AD-A085 586 AIR FORCE WRIGHT AERONAUTICAL LABS WRIGHT-PATTERSON AFB OH F/G 14/2
OPERATION OF GENERATOR TEST FACILITY.(U)
APR 80 P G GABERDIEL
UNCLASSIFIED AFWAL-TR-80-2033 NL

1 OF 1
AD A
085586

END
DATE
FILMED
7-80
DTIC

AFWAL-TR-80-2033

LEVEL

OPERATION OF GENERATOR TEST FACILITY

Power Systems Branch
Aerospace Power Division

April 1980

TECHNICAL REPORT AFWAL-TR-80-2033

Final Report for Period August 1979 to February 1980

Approved for public release; distribution unlimited.

AERO PROPULSION LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Philip G. Gaberdiel

PHILIP G. GABERDIEL, 1Lt, USAF
Project Engineer

Paul R. Bertheaud

PAUL R. BERTHEAUD
Technical Area Manager

FOR THE COMMANDER

J D Reams

JAMES D. REAMS
Chief, Aerospace Power Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/POOS-2*, W-PAFB, OH 45433 to help us maintain a current mailing list".*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

14

1. REPORT NUMBER: AFWAL-TR-80-2033
2. GOVT ACCESSION NO.: AD-A085586
3. RECIPIENT'S CATALOG NUMBER:

6

4. TITLE (and Subtitle): OPERATION OF GENERATOR TEST FACILITY.

9

5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report. Aug [redacted] 79-Feb [redacted] 80
6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Lt Philip G. Gaberdiel

10

8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Aero Propulsion Laboratory (POOS-2)
AF Wright Aeronautical Laboratories (AFSC)
Wright-Patterson Air Force Base, Ohio 45433

16

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Program Element 62203F, Project 3145, Task 314529, Work Unit 31452950
11. CONTROLLING OFFICE NAME AND ADDRESS:
Aero Propulsion Laboratory (POO)
AF Wright Aeronautical Laboratories (AFSC)
Wright-Patterson Air Force Base, Ohio 45433

11

12. REPORT DATE: Apr [redacted] 80

17

13. NUMBER OF PAGES: 90
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):

12 89

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Generator Test Facility

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This manual describes the operation of the Generator Test Facility of the Aero Propulsion Laboratory. This test facility is a computer-controlled facility for conducting performance tests of aircraft electrical generating systems.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

392662

FOREWORD

This report is an instruction manual for operating the Generator Test Facility. This test facility was built by engineers and technicians of the Power Systems Branch, Aerospace Power Division of the Aero Propulsion Laboratory, Wright-Patterson AFB, Ohio, under Project 3145, Task 314529, Work Unit 31452950. This manual was written by Lt. Philip G. Gaberdiel, AFWAL/POOS-2, during the period August 1979 through February 1980. The author submitted the report in February 1980.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION I

BACKGROUND

Present and future aircraft systems require that the aircraft electrical generation system provide high quality electrical power under a wide range of operating conditions. In order to determine the performance of prototype generating systems, the test engineer must conduct a series of tests in which the operating conditions of the generator are varied to simulate aircraft operation. Tests are performed with various electrical load levels presented to the generator, including overload and fault conditions. All tests require that the generator be operated at a known speed within its operating range. Some tests require the speed of the generator to be changed at specific rates of acceleration or deceleration. Other test conditions may be varied for particular generating systems. A generator test is defined by a specific set of these generator system operating conditions.

The test engineer must examine the output of the generator during a test in order to determine its performance in response to the conditions of the test. Measurements of the raw output of the generator must usually be processed by various analysis techniques in order to portray a clear measure of the performance of the test generator.

Conventional facilities for conducting generator tests suffer from several shortcomings. Control of test conditions such as generator speed and acceleration rate and load settings is cumbersome. Techniques for measuring generator output are often relatively inaccurate and are usually too slow to accurately measure high speed transients. Synchronization of test actions, especially data acquisition, is a major problem common to all manually operated facilities. In addition, most systems require that analysis of the test data be performed at another site which greatly increases analysis turnaround time.

The Generator Test Facility of the Aero Propulsion Laboratory (Figure 1) overcomes these shortcomings in the testing of aircraft electrical generating systems.

Computer control of the test facility provides a highly flexible mechanism for performing generator tests. The inherent systematic operation of a computer provides accurate synchronization of test actions. Computer control also requires a minimum of personnel to perform generator testing.

A high speed data acquisition system (Figure 2) provides an accurate digital representation of the output of the generator under test conditions. Software resident in the system computer performs the required analysis computations on the test data. These analysis results, which exhibit the transient response of the test generator, are then available for display to the test engineer. This analysis and display scheme greatly reduces the turnaround time required for data reduction.

Therefore, the Generator Test Facility enables the test engineer to conduct testing of aircraft electrical generating systems easily and accurately and to analyze the performance exhibited by the generator in response to the test conditions.

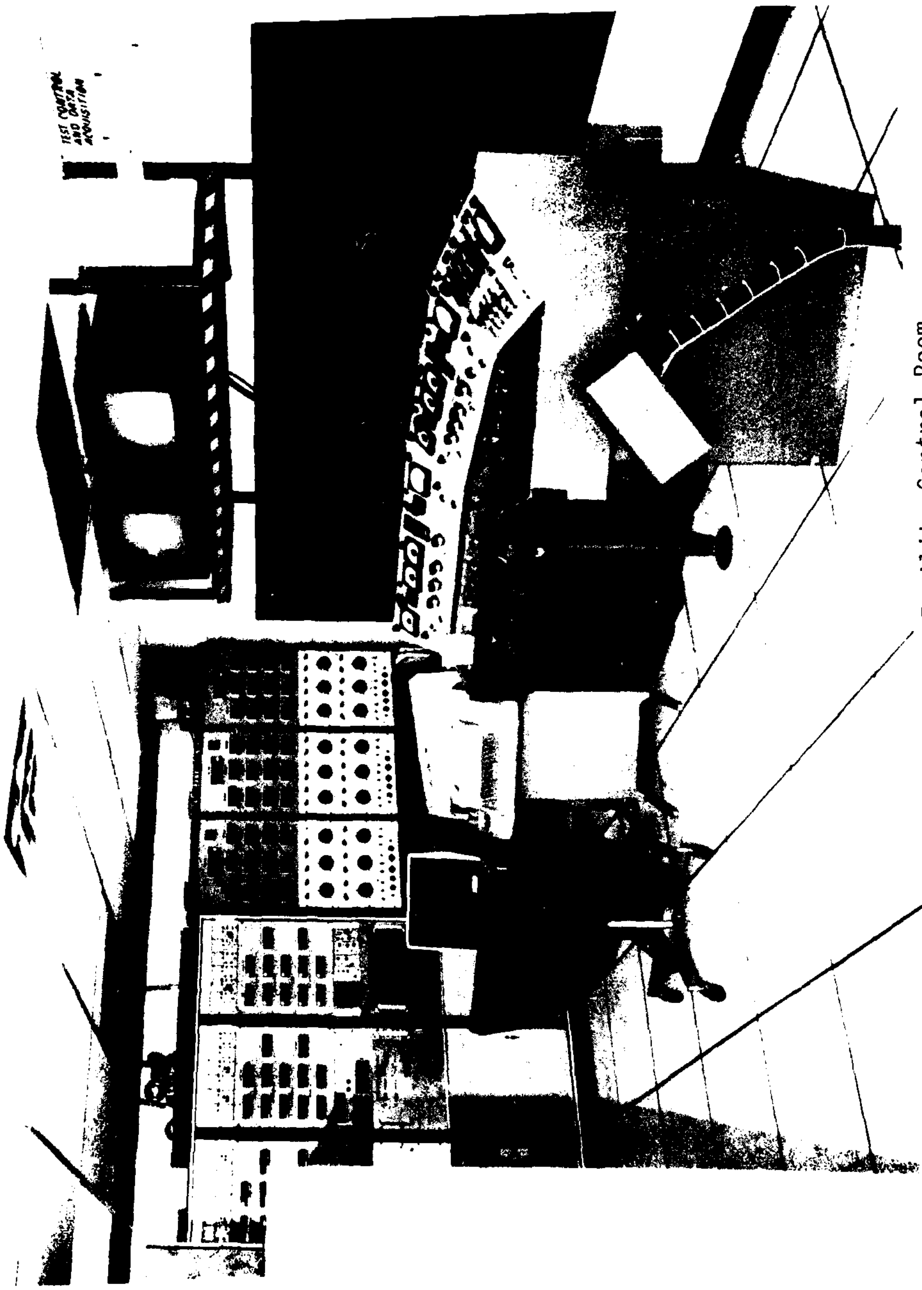

Figure 1. Generator Test Facility Control Room

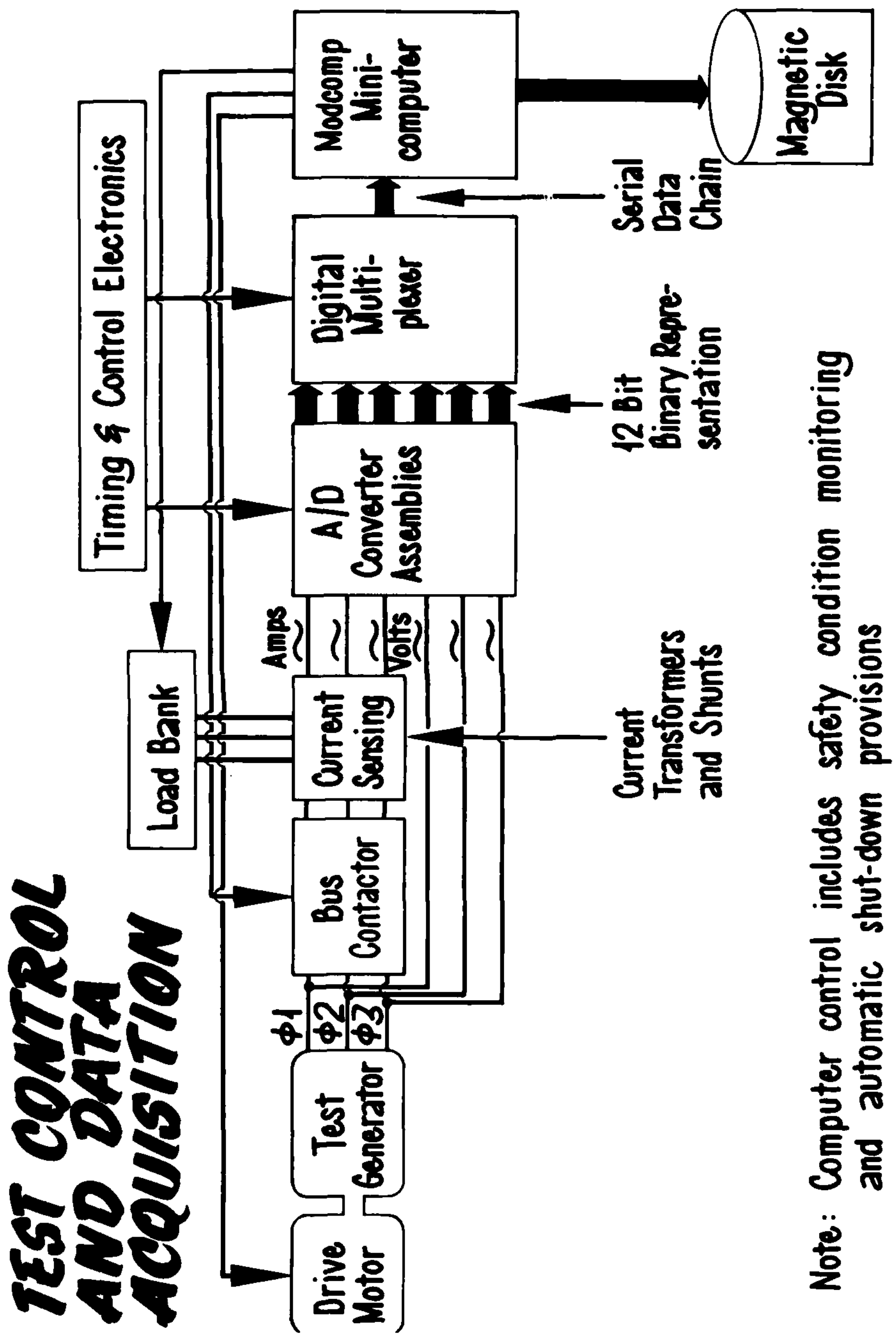

Figure 2. Test Control and Data Acquisition

SECTION II

INTRODUCTION

This manual describes the operation of the computer-controlled test capability of the Generator Test Facility. First discussed is the mechanism for creating the generator test sequences which will be used to direct operation of the generator system during a test. Next the manual explains the mechanisms, both software and hardware, which perform the test actions of a generator test sequence. And finally, the manual discusses operator actions required during computer-controlled testing.

The appendices include copies of all test control software required by the computer. Also included in the appendices are discussions about two special subsystems of the Generator Test Facility. These are the safety condition monitoring system and the drive stand controller.

This manual serves three major functions. First, it is a user's manual for the test engineer using the Generator Test Facility. Second, it can be used as a troubleshooting manual for the engineer or technician performing maintenance. Third, it serves as a training manual for personnel unfamiliar with the test facility. As such, the manual is written in general terms but with sufficient detail to serve the maintenance manual function.

SECTION III

CREATING A GENERATOR TEST SEQUENCE

A generator test sequence consists of a sequence of commands which control the test facility and the generator system during execution of a test. The commands available to the user are:

TM,	which controls relative timing of test actions;
DS,	which controls drive stand speed and acceleration rate;
LB,	which controls load bank settings;
AC,	which controls data acquisition; and
RC,	which controls various contact closure type functions.

By composing the appropriate sequence of these test commands, the test engineer can specify a particular generator test to be run.

The Fortran routine, CR8SYS, enables the user to compose and catalog generator test sequences. A listing of the code for CR8SYS is given in Appendix A. This section of the manual will give a description of each of the commands available for use in a test sequence. Also given is an example of the creation of a generator test sequence.

The creation of a test sequence is performed at the Tektronix CRT terminal. The routine CR8SYS, prompts the user to give the responses required to create the sequence. Each time the routine expects a new test sequence instruction, it will display the message, "ENTER COMMAND," on the CRT screen. The user will respond with a command and the parameters associated with that command. The user signals the end of the test sequence by entering the command "ST." The routine will then display the newly created test sequence on the ISC color display terminal. The user must then catalog the sequence so that it will be available for execution. Execution of a test sequence is explained in the next section of this manual. The following are the commands available for use in generator test sequences.

TM-Time

This command controls relative timing of events during the test sequence. It actually represents a delay until the time specified in its associated parameter. To specify a time delay, enter (via the Tektronix keyboard)

TM (time in seconds)*

The time value is a positive floating point number and must be greater than any previous time value used since the beginning of the test sequence.

As with all other commands requiring parameters, the TM command can be entered in two steps. If the user enters only "TM," the routine will prompt the user with the message "TIME (SEC)." The user then must enter the time in seconds of the delay required. This two-step entry is useful if the user is not familiar with the format of the parameters for the particular command chosen.

DS-Drive Stand Speed and Acceleration Rate

This command is used to select drive stand speed and acceleration/deceleration rate. The speed must be a positive integer between 0 and 10,000 rpm. The acceleration rate is a positive integer representing the rate in 100's of rpm per second at which the drive stand will accelerate from the previously requested speed (zero if no previous DS command exists in this sequence) to the speed requested by the current DS command. Therefore if the previous speed is greater than the requested speed, the rate actually represents deceleration.

The format of the command is:

DS (drive stand#) (speed) (rate)**

All parameters must be positive integers and are entered in free format with blanks or commas separating each one.

Again, if the user enters only "DS," the routine will respond with the prompt message, "DS#, SPEED, RATE (3I)." The user must then enter the appropriate parameters.

---

* All command entries must be completed by pressing the RETURN key.

** 3 is the minimum rate accepted by the routine, and 0 or 10 directs the drive stands to accelerate at the maximum rate of 1000 rpm/sec.

LB-Load Bank Settings

This command is used to select the setting of the load banks. Load banks numbered 1, 2, and 3 can be controlled. A load setting, both resistive (KW) and reactive (KVAR), is selected for each of three phases. The total load requested must not exceed the maximum available from the individual load bank. This would be 80 KW per phase and 60 KVAR per phase on load banks 1 or 2, and 40 KW per phase and 30 KVAR per phase on load bank 3.

The format of the command is:

LB (load bank#) (ϕ1KW) (ϕ2KW) (ϕ3KW) (ϕ1KVAR) (ϕ2KVAR) (ϕ3KVAR)

All parameters must be positive integers entered in free format.

If the user enters only "LB," the routine issues the prompt message, "LB#, 3*KW, 3*KVAR (7I)." The user must then enter the appropriate parameters.

AC-Acquire Data

This command causes data to be acquired from the data acquisition system for the amount of time specified in the parameter and stored on a dedicated magnetic disk file (file 30). Time spans of up to 25 seconds of data can be stored, but the analysis time required for this amount of data makes it practical to store time spans of 1 to 5 seconds of data.

The format of the command is:

AC (duration in seconds)

The duration value is a positive floating point number. If the user enters only "AC," the routine prompts with the message, "DURATION OF ACQUISITION (SEC)." Note that this command just initiates data acquisition and then proceeds to the next command. It does not wait for data acquisition to be completed before proceeding.

RC-Relay Contact Closure

This command is used to control opening or closing of contacts which are part of the input/output interface subsystem of the Modcomp minicomputer. These contacts can be used to control external relays used for such generator test functions as closing a bus contactor, applying generator field excitation, and the like. The contacts can accommodate loads of up to 10 VA with a maximum voltage of 100 V.

The user can control up to 13 contacts with this command. The desired state of the contact is specified as "0" or "1." A "0" opens the contact, and a "1" closes it.

The format of the command is:

RC (contact number) (state)

The parameters are positive integers entered in free format. If the user enters only "RC," the routine will prompt with the message, "CONTACT#, 1/0 (2I)." The user must then enter the required parameters.

There are also several commands to the routine CR8SYS which are used to maintain and catalog test sequences. These commands are explained below.

ST-Stop Sequence

This command signals the end of the test sequence being created. When the user responds with "ST," the routine displays the newly created test sequence on the ISC color display terminal. If correct, the sequence must be catalogued so that it will be available to be executed. If incorrect, the sequence must be reentered.

CA-Catalog Sequence

This command causes a test sequence to be stored on a library file of the magnetic disk (file 31). The test sequences are stored in a format such that the RD command can access them. There is storage available for sequences between 0 and 599 inclusive.

The format for this command is "CA." The routine will display the test sequence being stored on the ISC terminal and ask for "TEST SEQUENCE NUMBER, I5." The user enters the number at which he wishes the sequence to be catalogued. Any sequence previously catalogued at the same number will be overwritten. The sequence number is used for recalling the test sequence with the RD command.

RD-Read Test Sequence

This command allows the user to recall a test sequence from the magnetic disk (file 31). The sequence number requested must have been previously catalogued using the CA command.

The format of this command is:

RD (test sequence number)

If the user enters only "RD," the routine will prompt with the message, "ENTER TEST SEQUENCE NUMBER."

After the sequence number is entered, the routine will display the selected sequence on the ISC terminal. If the test sequence displayed has no instructions, this means that no sequence has been catalogued at the requested number.

SA-Read Same Sequence

This command causes the most recent sequence read or created to be displayed. The user chooses this command by typing "SA."

Following is a demonstration of the creation and cataloguing of a generator test sequence. All prompt messages produced by the routine and all responses required by the user are given.

This particular test sequence performs application of full load to a 150 KVA generator at base speed. The details of how these functions are accomplished during execution are given in the next section of this manual.

To create and store the test sequence, the following steps are required:

1. User types on the Tektronix keyboard

   JOB

   EXE CR8 LM

2. The display and responses illustrated in Figure 3 are via the Tektronix CRT terminal. Prompt messages issued by the routine are in all caps. Responses entered by the user are in brackets. A two-part entry of the AC command is included. Note: All responses by the user must be completed by striking the RETURN key.

3. At this point, the routine will produce on the ISC screen a display of the test sequence just created. Figure 4 illustrates that display.

4. The user must now catalog this sequence for later use. The following is the series of commands needed to do so.

```
ENTER COMMAND
[CA]
```

The routine will again display the sequence on the ISC screen and ask:

```
TEST SEQUENCE NUMBER, I5
[10]
ENTER COMMAND
$$ (This exits the CR8SYS routine.)
```

The newly created sequence is now catalogued as sequence number 10.

This section has demonstrated how to create and maintain generator test sequences using the routine CR8SYS. This section may be used as a user's manual and as an instructional guide for new personnel. Any modification to the sequence creation function of the routine CR8SYS will require a detailed examination of the Fortran code contained in Appendix A. The following section of this manual discusses the actual execution of a generator test sequence.

```
[JOB]
[EXE CR8 LM]
 ENTER COMMAND
[DS 3 4600 10]
 ENTER COMMAND
[TM 20.]
 ENTER COMMAND
[RC 5 1]
 ENTER COMMAND
[LB 2 50 50 50 0 0 0]
 ENTER COMMAND
[TM 24.8]
 ENTER COMMAND
[AC]
 DURATION OF ACQUISITION (SEC)
[1.5]
 ENTER COMMAND
[TM 25.]
 ENTER COMMAND
[RC 3 1]
 ENTER COMMAND
[TM 25.5]
 ENTER COMMAND
[RC 3 0]
 ENTER COMMAND
[TM 27.]
 ENTER COMMAND
[RC 4 1]
 ENTER COMMAND
[TM 27.5]
 ENTER COMMAND
[RC 4 0]
 ENTER COMMAND
[TM 30.]
 ENTER COMMAND
[RC 5 0]
 ENTER COMMAND
[LB 2 0 0 0 0 0 0]
 ENTER COMMAND
[DS 3 0 5]
 ENTER COMMAND
[ST]
 ENTER COMMAND
[CA]
 TEST SEQUENCE NUMBER, I5
[10]
 ENTER COMMAND
[$$]
```

Figure 3 - Test Sequence Creation Commands

TIME	DS	SPD	RATE	LB	KW			KVAR			DATA	RC	
	3	4600	10										
20.0												5	1
				2	50	50	50	0	0	0			
24.8											1.50		
25.0												3	1
25.5												3	0
27.0												4	1
27.5												4	0
30.0												5	0
				2	0	0	0	0	0	0			
	3	0	5										

Figure 4 - Test Sequence Display

SECTION IV

EXECUTING A GENERATOR TEST SEQUENCE

The Fortran routine RUNSYS directs the test facility during execution of a generator test sequence. A copy of the code for RUNSYS is included in Appendix B. The routine first allows the user to access a test sequence which was previously created and catalogued using the routine CR8SYS. The commands required to initiate execution of the routine RUNSYS are as follows: (via the Tektronix terminal).

```
[JOB]
[EXE RUN LM]
ENTER COMMAND
[RD 10]
```

The routine will display the requested test sequence (in this example, sequence 10) on the ISC terminal and ask,

```
READY TO RUN THIS SEQUENCE?  (Y OR N)
```

Section V will discuss actions required by the operator during execution of a test sequence.

The minicomputer directs execution of a generator test by causing appropriate actions to occur at specified times. These actions will be referred to as test control functions. Test control functions include: control of load bank settings, control of drive speed and acceleration rate, control of test data acquisition, and control of a set of relay contacts. A generator test is defined by a timed sequence for performing the test control functions required to accomplish the test. This timed sequence, referred to as a generator test sequence, is created using the test sequence creation function of the routine CR8SYS. The procedure for creating test sequences is given in Section III of this manual.

In order to cause the test defined in the test sequence to occur, a scheme must be devised which translates each command in the sequence

into an electrical output which performs the test function specified by the command. Generally this scheme is as follows: the Fortran routine RUNSYS steps through each command in the test sequence. For those functions which can be performed solely by software, RUNSYS calls the routine which implements the function. For those functions which are performed by software directing hardware interfaces, RUNSYS calls the routine which initiates action to perform the function. In some cases, several levels of control routines must be executed before the electrical signal is output to perform the function.

The test control functions used in generator testing are:

(1) Timing control

(2) Drive stand control

(3) Load bank control

(4) Data acquisition control

(5) Relay contact control

Following is a description of each function and the means by which this function is performed:

1. Timing control

In order to maintain proper sequencing of test actions during a generator test, the relative timing of the test control functions must be controlled. As explained in Section III, the TM command is used in the test sequence to specify this timing.

During test sequence execution, RUNSYS uses the subroutine TWAIT to provide timing control. A copy of TWAIT can be found in the code of RUNSYS in Appendix B.

At the beginning of execution of a test sequence, TWAIT is called to obtain a beginning time, TBEGIN, from the minicomputer internal timer. As TM commands are encountered during a test sequence,

TWAIT is called to provide a delay until the time specified in the command. By inserting TM commands just prior to other test control commands in a test sequence, the operator can specify the relative times at which events of the test occur.

2. Drive stand control

During generator testing, it is necessary to control the drive stand, which provides rotation to the generator, at a known speed. Also, some tests require the generator to be accelerated or decelerated to a new speed at a known rate. The drive stand control function provides this control.

The DS command is used in the test sequence to specify that one of the three drive stands be accelerated/decelerated to a specified speed at a specified rate. Actual control of the drive stand is accomplished by providing a DC voltage to the drive stand power supply. A voltage of 0-10V DC input to the power supply produces a speed of 0-10,000 rpm on the drive stand output pad. To control this speed, then, a DS command must be translated into the appropriate DC voltage. Control of acceleration rate is accomplished by outputting this DC voltage in small steps at an appropriate rate. For example, to accelerate from 1,000 rpm to 2,000 rpm at 2,000 rpm/second, a command to increase the speed by 20 rpm is issued every 10 msec for 0.5 seconds.

At this point, it is instructive to discuss the form in which certain control routines exist in the minicomputer. The routine RUNSYS is what is called a batch task. During execution of a test sequence, RUNSYS is executed by the minicomputer central processing unit (CPU) from its central memory. Certain test control functions, such as load bank control and relay contact control, can be performed by RUNSYS directly. However, other functions, namely, drive stand control and data acquisition control, are special in nature. These functions are iterative and must be performed with precise timing. For these reasons, the routines

which implement these functions exist as separate tasks. These tasks, named DSC for drive stand control and DAC for data acquisition control, share operating time in the minicomputer with the routine RUNSYS. It is useful to think of the DSC and DAC tasks as executing all the time. It should be clear that whenever a generator test is not being executed, the DSC task must output signals which keep the drive stands at zero speed and the DAC task must refrain from acquiring data.

The routine RUNSYS, when action is required from either of these tasks, collects the parameters from the appropriate test command, makes this data available to the task, and then signals the task to act on this data.

Explanation of how RUNSYS and the DSC task relate is now presented. A copy of the code for the DSC task is included in Appendix C. Discussion of the DAC task will be presented later in the appropriate section.

When RUNSYS encounters a DS command, it stores the drive stand number, requested speed, and requested rate in a buffer. It then sets a flag (DSC flag) which informs the DSC task that a valid command is available.

The DSC task has been executing a loop of instructions which output a null command to keep the drive stands at zero speed. On each iteration of this loop, the DSC flag is checked to see if a new command is available. When the DSC flag is found to be set, the DSC task first resets the flag so that it will know when a new command has been issued. Then it collects the command parameters from the buffer, scales and packs the command, and outputs the command to the drive stand controller. The DSC task continues to output this command until RUNSYS issues a new command.

As indicated above, the DSC task supplies its output to the drive stand controller. The DSC task, in a sense, interprets drive stand commands and passes these interpretations along to the controller.

The drive stand controller is a microprocessor based interface which implements drive stand commands. A digital-to-analog converter produces the DC voltage required by the drive stand power supply. The controller also performs the stepping increase/decrease described earlier to achieve acceleration rate control. A full description of the drive stand controller is given in Appendix E. By employing the controller to provide actual control of the drive stand, the DSC task must only initiate the action, thus leaving CPU time for RUNSYS and the DAC task.

The drive stand controller and the DSC task operate in a handshaking mode. If either device fails to respond correctly, the controller is designed to bring the drive stands to zero speed. Thus, if communication between the mini-computer and the drive stand controller is broken or if the minicomputer malfunctions, the drive stands are shut down.

3. Load bank control

The minicomputer controls the load bank settings during generator test execution by controlling a set of contacts in its input/output interface subsystem (I/OIS). There are 15 contacts for each phase of each load bank. Each contact, when closed, introduces a particular resistive or reactive load into the load circuit of the generator. Resistive loads up to 80 KW/phase and inductive loads up 60 KVAR/phase are available from loads banks #1 and #2. Loads up to 40 KW/phase and 30 KVAR/phase are available from load bank #3.

The computer control of the load bank settings parallels the manual control available via switch settings located on the load bank control panel. Therefore during execution of a generator test, these manual controls must be set to zero. Refer to the circuit drawings of the load bank control panel for physical location of these relays.

During the creation of a test sequence, the test operator specified load bank settings with the LB command. During execution of the

test sequence, the subroutine SETLB implements this control. SETLB examines the LB command parameters, sets up a control word which will close the appropriate relays to achieve the requested load settings, and then outputs this control word to the relay control system of the minicomputer. A listing of SETLB is included in the code of the routine RUNSYS in Appendix B.

4. Data acquisition control

The data acquired during a generator test consists of signals proportional to the 3-phase voltages and the 3-phase currents of the generator output. Data acquisition electronics convert these 6 analog signals to 12-bit digital representations. These binary words are multiplexed into data blocks which serve as input to the minicomputer through its 4805 data terminal. Each data block consists of 10 binary words: a null word, a timing word, 6 data words, and 2 data integrity words (parity and checksum).

In the creation of a test sequence, the user specifies the beginning time and duration of the data acquisition with the TM and AC commands. For instance, if the user were interested in a load application occurring 25 seconds into the test, he would specify at TM 24.8 seconds an AC command of duration 1.5 seconds.

During test execution when the routine RUNSYS encounters an AC command, it calls the subroutine ACQUIRE. Refer to the listing of RUNSYS in Appendix B for a copy of the code of ACQUIRE. The subroutine ACQUIRE first determines how many magnetic disk tracks will be required to store data for the duration requested. It then prepares buffer areas in central memory in which the data will be stored prior to being written to the disk. ACQUIRE also prepares data structures known as user file tables (UFT's) which are required in order to write to the disk. It then loads pointers to this information in a task communication area and resumes the data acquisition task (DAC) which has been in a hold state.

The DAC task performs the actual input and storage of generator test data. A copy of the code for the DAC task is included as Appendix D. In order to input data and write this data to the disk at a high rate, the DAC task employs a circular buffering technique. Nine temporary buffers in central memory are circularly filled with data incoming from the 4805. These are the buffers prepared by ACQUIRE. The transfer of data from the 4805 to these buffers is controlled by the direct memory processor (DMP) of the minicomputer. The central processor of the minicomputer is then directed by the DAC task to circularly output these buffers to the disk. A lockout scheme prevents overwriting a buffer before it has been written to disk or writing a buffer to disk before it is full. This lockout scheme reports to the user any attempted errors of this sort.

Nine buffers allow the data acquisition system to operate at approximately 9,200 hertz. This provides accurate sampling of up to 4,600 hertz input. A higher sampling rate or a change in the data word format will require additional buffering or a different buffering technique.

5. Relay contact control

The input/output interface subsystem (I/OIS) of the minicomputer contains relay contacts controllable by software. The relay contact control function enables the user to specify the state (on or off) of any of 13 of these contacts. Past test sequences have used these contacts to control such test actions as CSD disconnects, generator excitation voltage, etc.

The RC command is used in test sequence creation to specify the state of a particular relay. During test execution when RUNSYS encounters an RC command, it calls the subroutine SETRC. The code for SETRC can be found in the listing of RUNSYS in Appendix B. SETRC sets up a control word to properly set the requested contact and outputs it to the I/OIS.

This concludes the discussion of the test control functions available for generator testing. The next section of this manual discusses operator actions required during execution of a generator test.

test sequence, the subroutine SETLB implements this control. SETLB examines the LB command parameters, sets up a control word which will close the appropriate relays to achieve the requested load settings, and then outputs this control word to the relay control system of the minicomputer. A listing of SETLB is included in the code of the routine RUNSYS in Appendix B.

4. Data acquisition control

The data acquired during a generator test consists of signals proportional to the 3-phase voltages and the 3-phase currents of the generator output. Data acquisition electronics convert these 6 analog signals to 12-bit digital representations. These binary words are multiplexed into data blocks which serve as input to the minicomputer through its 4805 data terminal. Each data block consists of 10 binary words: a null word, a timing word, 6 data words, and 2 data integrity words (parity and checksum).

In the creation of a test sequence, the user specifies the beginning time and duration of the data acquisition with the TM and AC commands. For instance, if the user were interested in a load application occurring 25 seconds into the test, he would specify at TM 24.8 seconds an AC command of duration 1.5 seconds.

During test execution when the routine RUNSYS encounters an AC command, it calls the subroutine ACQUIRE. Refer to the listing of RUNSYS in Appendix B for a copy of the code of ACQUIRE. The subroutine ACQUIRE first determines how many magnetic disk tracks will be required to store data for the duration requested. It then prepares buffer areas in central memory in which the data will be stored prior to being written to the disk. ACQUIRE also prepares data structures known as user file tables (UFT's) which are required in order to write to the disk. It then loads pointers to this information in a task communication area and resumes the data acquisition task (DAC) which has been in a hold state.

SECTION V

TEST OPERATOR ACTIONS

Computer-controlled testing with the Generator Test Facility requires very little action by the test operator. Basically the test operator first ensures that the test facility and generator system are safe for operation; then the operator initiates the RUNSYS routine which will direct execution of the test. As detailed in Section IV, the steps required to initiate execution of RUNSYS are as follows: (via the Tektronix terminal)

[JOB]
[EXE RUN LM]
ENTER COMMAND
[RD n], where n is the test sequence number

The chosen sequence is displayed on the ISC terminal and the routine asks,

READY TO RUN THIS SEQUENCE? (Y OR N)

To initiate test execution, the user then strikes the "Y" and RETURN keys on the Tektronix keyboard. The routine RUNSYS then directs the test control actions as described in the previous section.

Actions required by the test operator during test sequence execution are of two basic categories. These are normal operation and abnormal operation of the test execution.

(1) Normal operation

Under normal operation, the operator performs the steps described above to initiate the test. The operator should continue to visually monitor the generator system during the test. If any unsafe condition arises, the operator should take actions to shut down the test.

In addition, the operator should note whether test actions (contact closures, load application and removal, etc.) occur at the expected time into the test. If these phenomena do not occur at approximately

the correct time, the test operator should take test shutdown actions since this is an indication that computer control of the test is not functioning properly. In order to monitor timing of the test, the test facility has a digital sequence timer. This timer should be started when execution of a test sequence is initiated.

Thus, under normal operation, the test operator should: (1) check the displayed test sequence for correctness, (2) visually inspect the generator and test area for safe operating conditions, (3) simultaneously start the digital timer and enter a "Y" (followed by RETURN) on the Tektronix keyboard, and then, (4) continue visual monitoring of the generator setup while checking that test phenomena occur at the proper time.

(2) Abnormal operation

The list of possible malfunctions which could occur to cause abnormal operation of a generator test is almost limitless. The following discussion will address several situations in which the generator test should be shut down. This shutdown action is required to prevent damage to the generator under test and possibly to the test facility or personnel.

Several types of malfunctions likely to occur during a generator test are automatically monitored by the minicomputer. This automatic monitoring is achieved in the following manner. First, a transducer develops a voltage signal proportional to the condition being monitored. In some cases, this voltage signal must be rectified, amplified, filtered, etc. to make it acceptable as input to the minicomputer. The conditional signal is then input to the mini-computer via its wide range analog input system. Then software routines check this data to determine if any of the signals are out of limit. The out-of-limit bounds are preselected for the particular condition being monitored and the particular test sequence. For example, a phase current of 600 amps would be an overcurrent condition under rated load testing of a generator, whereas the same current might be expected during fault testing.

If an out-of-limit condition is sensed, the minicomputer takes shutdown action automatically. A full discussion of the safety condition monitoring system is given in Appendix F. Also discussed in Appendix F is a safety monitoring routine available during manual operation of the test facility.

In some instances, the generator test should be shut down even if the safety monitoring system is not activated. These instances occur when the test operator observes any condition which he considers unsafe. For instance, suppose a test sequence specifies that the drive stand be accelerated to 4,600 rpm. During execution of the sequence, the operator observes the drive stand speed meter on the drive stand control console exceeding this speed. This indicates that computer control of the drive stands is not functioning properly. The operator should manually decelerate the drive. This is done by switching the control mode switch located on the drive stand control console to MANUAL and turning the DC motor lever to the READY position momentarily.

Another situation requiring manual shutdown is as follows. A test sequence specifies that a contact which is used to excite the test generator is to be closed 20 seconds after test initiation. During execution of the sequence, excitation does not occur. This situation in itself poses no danger, but it is evidence that the automatic sequencing of the test is not operating properly and more severe failures may occur. At the very least, without generator excitation, the data acquired during the test would be useless. Therefore, the operator should manually shut down the test.

Following is a list of the actions the test operator can take in shutting down a generator test.

1. In the event of a catastropic failure in which power should be removed from the drive stand as quickly as possible,	test operator should press the red MASTER EMERGENCY TRIP button on the center drive stand control console

2. In the event the drive stand is required to coast to a halt and assuming the drive motor electronics are functioning properly,	test operator should switch control mode switch to MANUAL and turn the DC MOTOR lever to the READY position.
3. To remove all electrical load from the generator,	test operator should set the master load switch on the load bank control console to OFF.

This concludes the discussion of operator actions during a computer-controlled generator test. The following section of this manual discusses actions required at the conclusion of a generator test sequence.

SECTION VI

CONCLUSION OF A GENERATOR TEST SEQUENCE

Every generator test sequence must conclude by restoring the computer-controlled functions of the test facility to their original states. For instance, all the load contacts in the load bank should be opened, the drive stands should be at zero speed, and all external relay contacts should be opened. The user must ensure that commands to perform these actions are included in the generator test sequence.

Thus, normal execution of a test sequence will reset all test functions. If the test is abnormally halted, either by the safety monitoring system or manually, the RUNSYS routine must be allowed to complete execution of the test sequence in order that all resetting is accomplished.

When the test has completed, the RUNSYS routine again issues the message, "ENTER COMMAND." At this point, the test data which was acquired during the test is stored on a dedicated file of the magnetic disk. By entering the following commands, the user initiates the data analysis and display routines stored in the minicomputer.

```
[EXE]
ENTER PROGRAM NAME; (A3)
[ANL]
```

These routines compute generator performance parameters and display the results to the test engineer via the Tektronix terminal. Details of the analysis procedures and use of the display routines are the subject of a separate instruction manual.

APPENDIX A
CR8SYS

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  09:30:56        PAGE   1

          PROGRAM CR8SYS
      C
      C  THIS ROUTINE ENABLES THE USER TO MAINTAIN GENERATOR
      C  TEST SEQUENCES.
      C  MAINTENANCE INCLUDES INITIAL CREATION, CATALOGUING, AND READING
      C  ALREADY CREATED SEQUENCES.
      C
      C  FILE/DEVICE ASSIGNMENTS REQUIRED FOR THIS ROUTINE:
      C
      C      8 = TKO (TEKTRONIX OUTPUT)  PROMPT MESSAGES TO USER
      C      7 = TKI (TEKTRONIX INPUT)   USER RESPONSES
      C      6 = A02 (ISC OUTPUT)        SEQUENCE DISPLAY
      C     31 = TSL (DISK PARTITION)    TEST SEQUENCE "LIBRARY"
      C
      C
      C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
      C  *  *  *  CATALOGUED ON LM 12,21,79   *  *  *  *  *  *  *  *
      C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
      C
      C  EXTERNAL SUBROUTINES REQUIRED:
      C
      C   NAME        LOCATION
      C
      C  2. READ         LB
      C  3. WRITE        LB
      C
      C
          INTEGER COMMAND(512),R(2),UFT(6),INUM,LINE(72)
          EQUIVALENCE (T,R)
      C  NOTE:  R IS AN INTEGER ARRAY WHICH IS EQUIVALENT TO THE REAL
      C  VARIABLE T.
      C
      C  INITIALIZE MAXIMUM NUMBER OF COMMANDS TO BE ALLOWED IN A TEST
      C  SEQUENCE
          DATA MAXCOM/512/
      C  SET UP UFT FOR WRITING/READING SEQUENCE TO DISK
      C  NOTE:  #BFE0 = @31 = TSL
          DATA UFT/0,ZBFE0,ZA400,3*0/
      C
      C  DESCRIPTION OF TEST SET-UP COMMANDS
      C
      C    TM - TIME DELAY
      C    DS - DRIVE STAND CONTROL
      C    LB - LOAD BANK CONTROL
      C    RC - RELAY CONTACT CONTROL
      C    AC - DATA ACQUISITION CONTROL
      C    ST - SPECIFIES END OF TEST SEQUENCE
      C
      C  MAINTENANCE COMMANDS:
      C
      C    CA - CATALOG SEQUENCE
      C    RD - READ SEQUENCE
      C    SA - REREAD SEQUENCE
      C
      C
      C  INITIALIZE POINTER INTO COMMAND BUFFER
```

*D MODCOMP SOURCE EDITOR   DATE 12/27/79 09:30:56   PAGE   2

```
100   IPNT=1
C  CHECK IF COMMAND BUFFER HAS BEEN EXCEEDED
110   IF(IPNT.LE.MAXCOM-8)GO TO 115
C  OUTPUT ERROR MESSAGE
      WRITE(8,1)
1     FORMAT(' COMMAND BUFFER EXCEEDED')
C  OUTPUT SEQUENCE AS IT STANDS
      GO TO 190
C  ISSUE PROMPT MESSAGE TO USER
115   WRITE(8,2)
2     FORMAT(' ENTER COMMAND')
C  INPUT COMMAND AND PARAMETERS
      READ(7,3,END=9999)ICOM,LINE
3     FORMAT(A2,72A1)
C  IS THIS A MAINTENANCE COMMAND?
      IF(ICOM.EQ.'SA')GO TO 190
C  REQUEST TO RE-CATALOG A SEQUENCE?
      IF(ICOM.EQ.'CA'.AND.IPNT.EQ.1)GO TO 190
C  IF NOT A MAINTENANCE COMMAND, ENTER INTO COMMAND BUFFER
      COMMAND(IPNT)=ICOM
C  DETERMINE WHICH COMMAND THIS IS AND GO TO ITS HANDLING ROUTINE
      IF(ICOM.EQ.'DS')GO TO 120
      IF(ICOM.EQ.'TM')GO TO 130
      IF(ICOM.EQ.'LB')GO TO 140
      IF(ICOM.EQ.'AC')GO TO 150
      IF(ICOM.EQ.'RC')GO TO 160
      IF(ICOM.EQ.'RD')GO TO 170
      IF(ICOM.EQ.'CA')GO TO 180
      IF(ICOM.EQ.'ST')GO TO 190
C  IF INVALID COMMAND, GO BACK AND GET NEW ONE
      WRITE(8,3000)
3000  FORMAT(' INVALID COMMAND')
      GO TO 110
C  ROUTINE TO INTERPRET DS COMMAND
C  IFORM COLLECTS INTEGER PARAMETERS FROM LINE AND STORES THEM IN
C  COMMAND BUFFER.  IERR INDICATES WHETHER PARAMETERS WERE VALID
C  INTEGERS.
120   CALL INTFORM(3,LINE,72,COMMAND(IPNT+1),IERR)
C  CHECK IF PARAMETERS WERE ENTERED
      IF(COMMAND(IPNT+1).EQ.0)GO TO 125
C  CHECK FOR INVALID PARAMETER
      IF(IERR.NE.0)GO TO 125
C  IF PARAMETERS WERE O.K., ADJUST POINTER
C  INTO COMMAND BUFFER
121   IPNT=IPNT+4
C  GET NEXT COMMAND
      GO TO 110
C  PROMPT USER TO ENTER PARAMETERS
125   WRITE(8,4)
4     FORMAT(' DS#,SPEED,RATE;(3I)')
C  INPUT FREE FORMAT PARAMETERS AND STORE IN COMMAND BUFFER
      CALL READI(7,COMMAND(IPNT+1))
C  ADJUST POINTER AND RETURN
      GO TO 121
C  ROUTINE TO INTERPRET TM COMMAND
C  COLLECT FLOATING POINT PARAMETER AND STORE IN T.  IERR RETURNS
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  09:30:56        PAGE   3

C  ERROR INDICATION.
130   CALL FPTFORM(LINE,72,T,IERR)
C  WAS PARAMETER ENTERED?
      IF(T.EQ.0.)GO TO 135
C  WAS PARAMETER VALID?
      IF(IERR.NE.0)GO TO 135
C  STORE FLOATING POINT PARAMETER AS TWO INTEGERS IN COMMAND BUFFER
132   COMMAND(IPNT+1)=R(1)
      COMMAND(IPNT+2)=R(2)
C  INCREMENT POINTER INTO COMMAND BUFFER
      IPNT=IPNT+3
C  GET NEXT COMMAND
      GO TO 110
C  PROMPT USER TO ENTER PARAMETER
135   WRITE(8,5)
5     FORMAT(' TIME(SEC)')
C  INPUT PARAMETER
      CALL READF(1,T)
C  STORE IN COMMAND BUFFER
      GO TO 132
C  ROUTINE TO INTERPRET LB COMMAND
C  COLLECT INTEGER PARAMETERS AND STORE IN COMMAND BUFFER
140   CALL INTFORM(7,LINE,72,COMMAND(IPNT+1),IERR)
C  CHECK FOR ERROR OR NO PARAMETERS
      IF(IERR.EQ.0.AND.COMMAND(IPNT+1).NE.0)GO TO 145
C  PROMPT USER TO ENTER PARAMETERS
      WRITE(8,6)
6     FORMAT(' LB#,3*KW,3*KVAR;(7I)')
C  COLLECT PARAMETERS AND STORE IN COMMAND BUFFER
      CALL READI(7,COMMAND(IPNT+1))
C  INCREMENT POINTER
145   IPNT=IPNT+8
C  GET NEXT COMMAND
      GO TO 110
C  ROUTINE TO INTERPRET AC COMMAND
C  COLLECT FLOATING POINT PARAMETER IN T
150   CALL FPTFORM(LINE,72,T,IERR)
C  CHECK FOR ERROR OR NO PARAMETER
      IF(IERR.EQ.0.AND.T.NE.0.)GO TO 155
C  PROMPT USER TO ENTER PARAMETERS
      WRITE(8,7)
7     FORMAT(' DURATION OF ACQUISITON(SEC)')
C  INPUT PARAMETER
      CALL READF(1,T)
C  STORE PARAMETER AS INTEGERS IN COMMAND BUFFER
155   COMMAND(IPNT+1)=R(1)
      COMMAND(IPNT+2)=R(2)
C  INCREMENT POINTER INTO COMMAND BUFFER
      IPNT=IPNT+3
C  GET NEXT COMMAND
      GO TO 110
C  ROUTINE TO INTERPRET RC COMMAND
C  COLLECT INTEGER PARAMETERS AND STORE IN COMMAND BUFFER
160   CALL INTFORM(2,LINE,72,COMMAND(IPNT+1),IERR)
C  CHECK FOR ERROR OR NO PARAMETERS
      IF(IERR.EQ.0.AND.COMMAND(IPNT+1).NE.0)GO TO 165
```

```
*D MODCOMP SOURCE EDITOR      DATE  12/27/79  29:30:56         PAGE   4

C   PROMPT USER TO ENTER PARAMETERS
        WRITE(8,8)
8       FORMAT(' CONTACT#,1/0;(2I)')
C   COLLECT PARAMETERS AND STORE IN COMMAND BUFFER
        CALL READI(2,COMMAND(IPNT+1))
C   INCREMENT POINTER
165     IPNT=IPNT+3
C   GET NEXT COMMAND
        GO TO 110
C   ROUTINE TO INTERPRET RD COMMAND
C   INPUT SEQUENCE NUMBER
170     CALL INTFORM(1,LINE,72,TNUM,IERR)
C   CHECK FOR ERROR OR NO RESPONSE
        IF(IERR.EQ.0.AND.TNUM.NE.0)GO TO 175
C   PROMPT USER TO ENTER SEQUENCE NUMBER
172     WRITE(8,9)
9       FORMAT(' ENTER TEST SEQUENCE NUMBER')
C   COLLECT SEQUENCE NUMBER
        CALL READI(1,TNUM)
C   CHECK FOR VALID SEQUENCE NUMBER
175     IF(TNUM.LT.0.OR.TNUM.GT.599)GO TO 172
C   SET RANDOM ADDRESS POINTER IN UFT
C   NOTE:  EACH SEQUENCE OCCUPIES 4 SECTORS
        UFT(4)=4*TNUM
C   READ SEQUENCE AND STORE IN COMMAND BUFFER
        DO 177 I=1,4
177     CALL READ(COMMAND((I-1)*128+1),256,UFT)
C   OUTPUT SEQUENCE IDENTIFIER
        WRITE(6,10)TNUM
10      FORMAT(//' TEST SEQUENCE NUMBER',I5)
C   OUTPUT SEQUENCE
        GO TO 190
C   ROUTINE TO HANDLE CA COMMAND
C   ENSURE THAT SEQUENCE ENDS WITH STOP COMMAND
180     COMMAND(IPNT)='ST'
C   OUTPUT SEQUENCE FOR USER TO VIEW
C   RESET POINTER
190     IPNT=1
C   OUTPUT HEADINGS
        WRITE(6,11)
11      FORMAT('    TIME',3X,'DS',2X,'SPD',2X,'RATE',3X,'LB',6X,'KW',
       19X,'KVAR',8X,'DATA',7X,'RC'//)
C   OUTPUT COMMANDS, ONE AT A TIME
200     NCMD=COMMAND(IPNT)
        IF(NCMD.EQ.'DS')GO TO 201
        IF(NCMD.EQ.'TM')GO TO 202
        IF(NCMD.EQ.'LB')GO TO 203
        IF(NCMD.EQ.'AC')GO TO 204
        IF(NCMD.EQ.'RC')GO TO 205
        IF(NCMD.EQ.'ST')GO TO 206
C   IF NONE OF THESE, ASSUME ST
        GO TO 206
C   OUTPUT DS COMMAND PARAMETERS
201     WRITE(6,12)(COMMAND(IPNT+I),I=1,3)
12      FORMAT(10X,I1,2I6)
C   UPDATE POINTER
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  09:30:56        PAGE   5

            IPNT=IPNT+4
      C  GET NEXT COMMAND
            GO TO 200
      C  OUTPUT TM COMMAND PARAMETER
      C  COLLECT REAL PARAMETER INTO INTEGER ARRAY EQUIVALENCED TO
      C  REAL VARIABLE
      202   R(1)=COMMAND(IPNT+1)
            R(2)=COMMAND(IPNT+2)
      C  OUTPUT PARAMETER
            WRITE(6,13)T
      13    FORMAT(1X,F6.1)
      C  UPDATE POINTER
            IPNT=IPNT+3
      C  GET NEXT COMMAND
            GO TO 200
      C  OUTPUT LB COMMAND PARAMETERS
      203   WRITE(6,14)(COMMAND(IPNT+I),I=1,7)
      14    FORMAT(25X,I2,1X,6I4)
      C  UPDATE POINTER
            IPNT=IPNT+8
      C  GET NEXT COMMAND
            GO TO 200
      C  OUTPUT AC COMMAND PARAMETER
      C  COLLECT REAL PARAMETER INTO INTEGER ARRAY EQUIVALENCED TO
      C  REAL VARIABLE
      204   R(1)=COMMAND(IPNT+1)
            R(2)=COMMAND(IPNT+2)
      C  OUTPUT PARAMETER
            WRITE(6,15)T
      15    FORMAT(55X,F6.2)
      C  UPDATE POINTER
            IPNT=IPNT+3
      C  GET NEXT COMMAND
            GO TO 200
      C  OUTPUT RC COMMAND PARAMETERS
      205   WRITE(6,16)(COMMAND(IPNT+I),I=1,2)
      16    FORMAT(65X,2I3)
      C  UPDATE POINTER
            IPNT=IPNT+3
      C  GET NEXT COMMAND
            GO TO 200
      C  CHECK TO SEE IF THIS SEQUENCE IS TO BE CATALOGUED
      206   IF(ICOM.NE.'CA')GO TO 100
      C  PROMPT USER TO ENTER SEQUENCE NUMBER TO CATALOG UNDER
      210   WRITE(8,17)
      17    FORMAT(' ENTER TEST SEQUENCE NUMBER')
      C  INPUT SEQUENCE NUMBER
            CALL READI(1,I)
      C  CHECK VALIDITY OF SEQUENCE NUMBER
            IF(I.LT.0.OR.I.GT.599)GO TO 210
      C  SET RANDOM ADDRESS POINTER IN UFT
            UFT(4)=4*I
      C  OUTPUT SEQUENCE TO DISK AS 4 SECTORS
            DO 220 I=1,4
      220   CALL WRITE(COMMAND((I-1)*128+1),256,UFT)
      C  GET NEXT COMMAND
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  09:30:56        PAGE   6

          GO TO 100
    9999  STOP
          END
    C
    C
    C  THE FOLLOWING ROUTINES ARE USED DURING TEST SEQUENCE CREATION
    C  TO INPUT FREE-FORMAT REAL AND INTEGER PARAMETERS FOR
    C  GENERATOR TEST COMMANDS.
    C
    C
          SUBROUTINE READI(N,IARRAY)
    C
    C  ROUTINE INPUTS ASCII STRING IN FREE FORMAT AND RETURNS N
    C  EQUIVALENT INTEGER VALUES IN IARRAY
    C
          INTEGER IARRAY(1),BUF(74),BUFSIZ
          DATA BUFSIZ/74/
    C  INPUT STRING OF ASCII CHARACTERS
    100   READ(7,1000)BUF
    1000  FORMAT(74A1)
    C  CALL ROUTINE TO FORMAT AND CONVERT TO INTEGER
          CALL INTFORM(N,BUF,BUFSIZE,IARRAY,IERR)
    C  CHECK FOR ERROR IN PARAMETERS
          IF(IERR.EQ.0)RETURN
    C  ECHO INVALID CHARACTER TO USER
          WRITE(8,2000)BUF(IERR)
    2000  FORMAT(' INVALID CHARACTER -',A1,5X,'RE-ENTER LINE')
    C  GET NEW STRING
          GO TO 100
          END
    C
    C
          SUBROUTINE INTFORM(NPAR,STRING,LTH,INT,IERR)
    C
    C  ROUTINE CONVERTS ASCII STRING (STRING) OF LENGTH, LTH, TO
    C  NPAR EQUIVALENT INTEGERS STORED IN ARRAY INT.  IERR IS SET
    C  IF ANY CHARACTER IN STRING IS NOT A VALID DIGIT.
    C
          INTEGER INT(1),STRING(1)
    C  INITIALIZE ERROR FLAG
          IERR=0
    C  INITIALIZE POINTER INTO STRING
          IPNT=1
    C  CONVERT EACH INTEGER
          DO 100 I=1,NPAR
    C  INITIALIZE VALUE
          INT(I)=0
    C  CHECK IF POINTER EXCEEDS STRING LENGTH
    10    IF(IPNT.GT.LTH)GO TO 100
    C  CHECK FOR DELIMITER
    20    IF(STRING(IPNT).NE.' '.AND.STRING(IPNT).NE.',')GO TO 30
    C  INCREMENT POINTER
          IPNT=IPNT+1
    C  GET NEXT CHARACTER
          GO TO 10
    C  CHECK FOR VALID DIGIT
```

```
** MODCOMP SOURCE EDITOR      DATE  12/27/79  09:30:56         PAGE   7

30     IF(STRING(IPNT).LE.'9'.AND.STRING(IPNT).GE.'0')GO TO 40
C   SET ERROR FLAG AND RETURN
       IERR=IPNT
       RETURN
C   CONVERT TO INTEGER AND ADD TO ACCUMULATION OF VALUE
40     INT(I)=INT(I)*10 + (STRING(IPNT)-'0')/256
C   INCREMENT POINTER
       IPNT=IPNT+1
C   CHECK FOR END OF STRING OR DELIMITER
       IF(IPNT.LE.LTH.AND.STRING(IPNT).NE.' '.AND.
      1STRING(IPNT).NE.',')GO TO 30
100    CONTINUE
       RETURN
       END
C
C
       SUBROUTINE READF(N,ARRAY)
C
C   ROUTINE INPUTS ASCII STRING IN FREE-FORMAT AND RETURNS N
C   EQUIVALENT FLOATING POINT VALUES IN ARRAY.
C   NOTE:  IN ALL CURRENT CALLS TO READF, ONLY 1 VALUE IS
C   INPUT. (I.E., N=1)
C
       INTEGER BUF(74),BUFSIZ
       REAL ARRAY(1)
       DATA BUFSIZ/74/
C   INPUT STRING OF ASCII CHARACTERS
100    READ(7,1000)BUF
1000   FORMAT(74A1)
C   CALL ROUTINE TO FORMAT AND CONVERT TO FLOATING POINT
       CALL FPTFORM(BUF,BUFSIZ,ARRAY,IERR)
C   CHECK FOR ERRORS
       IF(IERR.EQ.0)RETURN
C   ECHO INVALID CHARACTER TO USER
       WRITE(8,2000)BUF(IERR)
2000   FORMAT(' INVALID CHARACTER -',A1,5X,'RE-ENTER LINE')
C   GET NEW STRING
       GO TO 100
       END
C
C
       SUBROUTINE FPTFORM(STRING,LTH,FPT,IERR)
C
C   ROUTINE CONVERTS ASCII STRING (STRING) OF LENGTH, LTH, TO
C   EQUIVALENT FLOATING POINT VARIABLE, FPT.
C   IERR IS SET IF ANY CHARACTER IN STRING IS NOT A VALID DIGIT
C   OR DECIMAL POINT.
C
       INTEGER STRING(1),PFLAG,PAMT
C   INITIALIZE ERROR FLAG
       IERR=0
C   INITIALIZE POINTER INTO STRING
       IPNT=1
C   INITIALIZE RETURN VALUE
       FPT=0.
C   INITIALIZE DECIMAL POINT INDICATOR
```

*D MODCOMP SOURCE EDITOR DATE 12/27/79 09:30:56 PAGE 8

```
      PFLAG=0
C  PAMT KEEPS TRACK OF HOW MANY DIGITS OCCUR AFTER DECIMAL POINT
      PAMT=0
C  CHECK IF ALL OF STRING HAS BEEN EXAMINED
100   IF(IPNT.GT.LTH)RETURN
C  CHECK FOR DELIMITER
      IF(STRING(IPNT).NE.' '.AND.STRING(IPNT).NE.',')GO TO 110
C  INCREMENT POINTER
      IPNT=IPNT+1
C  CHECK NEXT CHARACTER
      GO TO 100
C  CHECK FOR DECIMAL POINT
110   IF(STRING(IPNT).NE.'.')GO TO 120
C  SET INDICATOR
      PFLAG=1
C  GET NEXT CHARACTER
      GO TO 140
C  CHECK FOR VALID DIGIT
120   IF(STRING(IPNT).LE.'9'.AND.STRING(IPNT).GE.'0')GO TO 130
C  SET ERROR FLAG AND RETURN
      IERR=IPNT
      RETURN
C  CHANGE ASCII DIGIT TO FLOATING POINT AND ADD TO ACCUMULATION
130   FPT=FPT*10. + FLOAT((STRING(IPNT)-'0')/256)
C  UPDATE NUMBER OF DIGITS AFTER DECIMAL POINT
      PAMT=PAMT+PFLAG
C  INCREMENT POINTER
140   IPNT=IPNT+1
C  CHECK FOR END OF STRING OR DELIMITER
      IF(IPNT.GT.LTH.OR.STRING(IPNT).EQ.' '.OR.STRING(IPNT).EQ.
     1',')GO TO 150
C  CHECK NEXT CHARACTER
      GO TO 110
C  ADJUST FPT TO ACCOUNT FOR DECIMAL POINT
150   FPT=FPT*10.**(-PAMT)
C  RETURN TO MAIN ROUTINE
      RETURN
      END
```

APPENDIX B
RUNSYS

*D MODCOMP SOURCE EDITOR   DATE 12/27/79 09:30:20   PAGE   1

```
      PROGRAM RUNSYS
C
C  THIS ROUTINE PERFORMS EXECUTION OF A SEQUENCE SELECTED
C  BY THE USER.  EXECUTION OF A TEST SEQUENCE IS
C  ACCOMPLISHED BY SEQUENTIALLY INVOKING THE ROUTINE OR TASK
C  REQUIRED TO PERFORM EACH COMMAND IN THE SEQUENCE.
C
C  FILE/DEVICE ASSIGNMENTS REQUIRED FOR THIS ROUTINE:
C
C     8 = TKO (TEKTRONIX OUTPUT)  PROMPT MESSAGES TO USER
C     7 = TKI (TEKTRONIX INPUT)   USER RESPONSES
C     6 = A02 (ISC OUTPUT)        SEQUENCE DISPLAY
C    31 = TSL (DISK PARTITION)    TEST SEQUENCE "LIBRARY"
C
C
C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C  *  *  *  CATALOGUED ON LM 12,26,79   *  *  *  *  *  *  *  *
C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C
C  EXTERNAL SUBROUTINES REQUIRED:
C
C   NAME       LOCATION
C
C  1. TIMER       UL
C  2. READ        LB
C  3. WRITE       LB
C  4. INITSAF     UL
C  5. TSKDEL      UL
C  6. SETBF       UL
C  7. SETBT       UL
C  8. GETDATA     UL
C
C
      INTEGER COMMAND(512),R(2),UFT(6),PNAME(2),TNUM,LINE(72)
      EQUIVALENCE (T,R)
C  NOTE:  R IS AN INTEGER ARRAY WHICH IS EQUIVALENT TO THE REAL
C  VARIABLE T.
C
C  INITIALIZE MAXIMUM NUMBER OF COMMANDS TO BE ALLOWED IN A TEST
C  SEQUENCE
      DATA MAXCOM/512/
C  SET UP UFT FOR WRITING/READING SEQUENCE TO DISK
C  NOTE:  #BFE0 = @31 = TSL
      DATA UFT/0,ZBFE0,ZA400,3*0/
C
C  DESCRIPTION OF TEST COMMANDS
C
C    TM - TIME DELAY
C    DS - DRIVE STAND CONTROL
C    LB - LOAD BANK CONTROL
C    RC - RELAY CONTACT CONTROL
C    AC - DATA ACQUISITION CONTROL
C    ST - SPECIFIES END OF TEST SEQUENCE
C
C  MAINTENANCE COMMANDS:
C
```

```
C     RD - READ SEQUENCE
C     SA - REREAD SEQUENCE
C     EX - EXECUTE OVERLAY FROM LM FILE
C
C
C   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C   *   *   *   SELECTION OF TEST SEQUENCE   *   *   *   *   *   *
C   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C
C  INITIALIZE POINTER INTO COMMAND BUFFER
100   IPNT=1
C  ISSUE PROMPT MESSAGE TO USER
115   WRITE(8,2)
2     FORMAT(' ENTER COMMAND')
C  INPUT COMMAND AND PARAMETERS
      READ(7,3,END=9999)ICOM,LINE
3     FORMAT(A2,72A1)
C  IS THIS A MAINTENANCE COMMAND?
      IF(ICOM.EQ.'SA')GO TO 190
C  IF NOT A MAINTENANCE COMMAND, ENTER INTO COMMAND BUFFER
      COMMAND(IPNT)=ICOM
C  DETERMINE WHICH COMMAND THIS IS AND GO TO ITS HANDLING ROUTINE
      IF(ICOM.EQ.'RD')GO TO 170
      IF(ICOM.EQ.'EX')GO TO 300
C  IF INVALID COMMAND, GO BACK AND GET NEW ONE
      WRITE(8,2000)
2000  FORMAT(' INVALID COMMAND')
      GO TO 115
C  ROUTINE TO INTERPRET RD COMMAND
C  INPUT SEQUENCE NUMBER
170   CALL INTFORM(1,LINE,72,TNUM,IERR)
C  CHECK FOR ERROR OR NO RESPONSE
      IF(IERR.EQ.0.AND.TNUM.NE.0)GO TO 175
C  PROMPT USER TO ENTER SEQUENCE NUMBER
172   WRITE(8,9)
9     FORMAT(' ENTER TEST SEQUENCE NUMBER')
C  COLLECT SEQUENCE NUMBER
      CALL READI(1,TNUM)
C  CHECK FOR VALID SEQUENCE NUMBER
175   IF(TNUM.LT.0.OR.TNUM.GT.599)GO TO 172
C  SET RANDOM ADDRESS POINTER IN UFT
C  NOTE:  EACH SEQUENCE OCCUPIES 4 SECTORS
      UFT(4)=4*TNUM
C  READ SEQUENCE AND STORE IN COMMAND BUFFER
      DO 177 I=1,4
177   CALL READ(COMMAND((I-1)*128+1),256,UFT)
C  OUTPUT SEQUENCE IDENTIFIER
      WRITE(6,10)TNUM
10    FORMAT(///' TEST SEQUENCE NUMBER',I5)
C  OUTPUT SEQUENCE FOR USER TO VIEW
C  RESET POINTER
190   IPNT=1
C  OUTPUT READINGS
      WRITE(6,11)
11    FORMAT('   TIME',3X,'DS',2X,'SPD',2X,'RATE',3X,'LB',6X,'KW',
     19X,'KVAR',8X,'DATA',7X,'RC'//)
```

```
C  OUTPUT COMMANDS, ONE AT A TIME
200    NCMD=COMMAND(IPNT)
       IF(NCMD.EQ.'DS')GO TO 201
       IF(NCMD.EQ.'TM')GO TO 202
       IF(NCMD.EQ.'LB')GO TO 203
       IF(NCMD.EQ.'AC')GO TO 204
       IF(NCMD.EQ.'RC')GO TO 205
       IF(NCMD.EQ.'ST')GO TO 900
C  IF NONE OF THESE, ASSUME ST
       GO TO 900
C  OUTPUT DS COMMAND PARAMETERS
201    WRITE(6,12)(COMMAND(IPNT+I),I=1,3)
12     FORMAT(10X,I1,2I6)
C  UPDATE POINTER
       IPNT=IPNT+4
C  GET NEXT COMMAND
       GO TO 200
C  OUTPUT TM COMMAND PARAMETER
C  COLLECT REAL PARAMETER INTO INTEGER ARRAY EQUIVALENCED TO
C  REAL VARIABLE
202    R(1)=COMMAND(IPNT+1)
       R(2)=COMMAND(IPNT+2)
C  OUTPUT PARAMETER
       WRITE(6,13)T
13     FORMAT(1X,F6.1)
C  UPDATE POINTER
       IPNT=IPNT+3
C  GET NEXT COMMAND
       GO TO 200
C  OUTPUT LB COMMAND PARAMETERS
203    WRITE(6,14)(COMMAND(IPNT+I),I=1,7)
14     FORMAT(25X,I2,1X,6I4)
C  UPDATE POINTER
       IPNT=IPNT+8
C  GET NEXT COMMAND
       GO TO 200
C  OUTPUT AC COMMAND PARAMETER
C  COLLECT REAL PARAMETER INTO INTEGER ARRAY EQUIVALENCED TO
C  REAL VARIABLE
204    R(1)=COMMAND(IPNT+1)
       R(2)=COMMAND(IPNT+2)
C  OUTPUT PARAMETER
       WRITE(6,15)T
15     FORMAT(55X,F6.2)
C  UPDATE POINTER
       IPNT=IPNT+3
C  GET NEXT COMMAND
       GO TO 200
C  OUTPUT RC COMMAND PARAMETERS
205    WRITE(6,16)(COMMAND(IPNT+I),I=1,2)
16     FORMAT(65X,2I3)
C  UPDATE POINTER
       IPNT=IPNT+3
C  GET NEXT COMMAND
       GO TO 200
C  ROUTINE TO LOAD OVERLAY FROM LM FILE.
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  09:30:20        PAGE   4

C  NOTE:  LOADING OVERLAY TRANSFERS CONTROL FROM RUNSYS TO THE
C  ROUTINE OVERLAYED.
C  ONLY USE OF THIS FEATURE SO FAR HAS BEEN TO LOAD ANALYSIS
C  ROUTINE AT THE COMPLETION OF A GENERATOR TEST SEQUENCE.
C
C  PROMPT USER TO ENTER OVERLAY NAME
300    WRITE(8,18)
18     FORMAT(' ENTER PROGRAM NAME;(A3)')
C  INPUT NAME
       READ(7,19)PNAME
19     FORMAT(2A2)
C  ROUTINE WHICH FOLLOWS PERFORMS OVERLAY OF REQUESTED ROUTINE
       INLINE
         LDM,2  PNAME
         LDM,3  PNAME+1      GET ROUTINE NAME
         REX,#37             CONVERT TO CAN CODE
         DFC   $+1
         STM,3  PNAM         STORE IN PNAM
         REX,#43             GET INFO ON RUNSYS
         DFC   0
         TRR,2,3             REG 2=START ADDR OF RUNSYS
         REX,#2B             LOAD CHAIN OVERLAY
         DFC   0,OLM
PNAM     RES   1        STORAGE FOR PROGRAM NAME
         DFC   0
       FINI
C  NOTE:  AT THIS POINT DURING EXECUTION, CONTROL HAS PASSED TO
C  THE OVERLAY.
C
C
C  ASK USER IF THIS SEQUENCE IS TO BE EXECUTED
900    WRITE(8,20)
20     FORMAT('0READY TO RUN THIS SEQUENCE?(Y OR N)')
C  INPUT RESPONSE
       READ(7,21)ICH
21     FORMAT(A2)
C  CHECK RESPONSE; IF NOT Y, GET NEXT COMMAND
       IF(ICH.NE.'Y')GO TO 100
C
C   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C   BEGINNING OF TEST SEQUENCE EXECUTION
C   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C
C  INITIATE SAFETY MONITORING PERFORMED IN DSC TASK.
C  NOTE:  ITEMP IS STORAGE LOCATION USED BY ROUTINE INITSAF TO
C  SAVE SYSTEM SI POINTER
       ITEMP=0
       CALL INITSAF(ITEMP)
C  INITIALIZE POINTER INTO COMMAND BUFFER
       IPNT=1
C  OBTAIN SYSTEM CLOCK READING AT START OF TEST
       CALL TWAIT(0.)
C  CHECK IF POINTER EXCEEDS BUFFER
1000   IF(IPNT.GE.MAXCOM)GO TO 100
C  EXAMINE EACH COMMAND OF SEQUENCE AND JUMP TO INSTRUCTIONS
C  WHICH WILL INITIATE ACTION REQUIRED BY THAT COMMAND.
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  09:30:20        PAGE   5

       NCMD=COMMAND(IPNT)
       IF(NCMD.EQ.'DS')GO TO 1100
       IF(NCMD.EQ.'TM')GO TO 1200
       IF(NCMD.EQ.'LB')GO TO 1300
       IF(NCMD.EQ.'AC')GO TO 1400
       IF(NCMD.EQ.'RC')GO TO 1500
C  IF NONE OF THESE COMMANDS, TEST SEQUENCE IS THROUGH
C  TERMINATE INPUT OF SAFETY MONITORING PARAMETERS AND RELOAD
C  ORIGINAL SI POINTER
       INLINE
         LDI,3  #4400     TERMINATE COMMAND
         OCB,3,2          ISSUE COMMAND TO OUTPUT DEVICE
         OCB,3,3          ISSUE COMMAND TO INPUT DEVICE
         LDM,10  ITEMP    GET SI POINTER
         STM,10  #00D3    STORE IN SYSTEM SI TRAP
         LDI,2  @SBF      SBF FLAG
         ZRR,3
         REX,#4E          ZERO FLAG
       FINI
       GO TO 100
C  ROUTINE WHICH PASSES DS PARAMETERS TO SUBROUTINE WHICH WILL
C  INITIATE DRIVE STAND CONTROL
1100  CALL SETDS(COMMAND(IPNT+1))
C  INCREMENT POINTER INTO COMMAND BUFFER
       IPNT=IPNT+4
C  GET NEXT COMMAND
       GO TO 1000
C  CALL ROUTINE TO PROVIDE REQUESTED DELAY
1200  CALL TWAIT(COMMAND(IPNT+1))
C  INCREMENT POINTER
       IPNT=IPNT+3
C  GET NEXT COMMAND
       GO TO 1000
C  PASS LB PARAMETERS TO SUBROUTINE WHICH CONTROLS LOAD BANK
C  SETTINGS
1300  CALL SETLB(COMMAND(IPNT+1),COMMAND(IPNT+2),COMMAND(IPNT+5)
     1,IERR)
C  INCREMENT POINTER
       IPNT=IPNT+8
C  GET NEXT COMMAND
       GO TO 1000
C  CALL ROUTINE TO INITIATE DATA ACQUISITION
1400  CALL ACQUIRE(COMMAND(IPNT+1))
C  INCREMENT POINTER
       IPNT=IPNT+3
C  GET NEXT COMMAND
       GO TO 1000
C  CALL ROUTINE WHICH CONTROLS RELAY CONTACTS
1500  CALL SETRC(COMMAND(IPNT+1),IERR)
C  INCREMENT POINTER
       IPNT=IPNT+3
C  GET NEXT COMMAND
       GO TO 1000
C
C   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C   END OF TEST SEQUENCE EXECUTION
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  09:30:20        PAGE   6

C   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C
9999  STOP
      END
C
C
C  FOLLOWING ROUTINES INITIATE ACTION REQUIRED BY INDIVIDUAL
C  COMMANDS IN TEST SEQUENCE.
C
C
      SUBROUTINE SETDS(COMMAND)
C
C  THIS ROUTINE MAKES PARAMETERS OF DS COMMAND AVAILABLE TO
C  DRIVE STAND CONTROL TASK, DSC.
C
      INTEGER COMMAND(1),COM(3)
C  GET PARAMETERS IN ARRAY ACCESSIBLE TO INLINE CODE
      DO 10 I=1,3
10    COM(I)=COMMAND(I)
C  PLACE POINTER TO PARAMETERS IN TASK COMMUNICATION AREA
      INLINE
        LDI,2  @DSC       POINTER IS NAMED DSC
        LDI,3  COM
        REX,#4E           STORE POINTER
      FINI
      RETURN
      END
C
C
      SUBROUTINE TWAIT(T)
C
C  ROUTINE DELAYS EXECUTION OF RUNSYS (I.E., OPERATION OF
C  GENERATOR TEST) UNTIL SPECIFIED TIME, T.  THIS DELAY PROVIDES
C  TIMING CONTROL OF TEST SEQUENCE.
C
      DOUBLE PRECISION TBEGIN,TNOW
C  IF REQUESTED TIME EQUALS ZERO, SAVE CURRENT SYSTEM CLOCK READING
      IF(T.NE.0.)GO TO 1
      CALL TIMER(TBEGIN)
      RETURN
1     CONTINUE
C  GET SYSTEM CLOCK READING
      CALL TIMER(TNOW)
C  IS DELAY POSITIVE?
      IF((DBLE(T)+TBEGIN-.025D0).GT.TNOW)GO TO 2
C  NOTIFY USER OF TIMING ERROR
      WRITE(6,100)T
100   FORMAT(' TIME ERROR',E15.5)
      RETURN
C  CONVERT TIME IN SECONDS TO MINUTES AND TICS
2     MIN=IFIX(SNGL((TBEGIN+DBLE(T))/60.D0))
      ITIC=IFIX(SNGL(TBEGIN+DBLE(T)-60.D0*DBLE(FLOAT(MIN)))*200.)
C  CALL SYSTEM ROUTINE TO DELAY UNTIL REQUESTED TIME
      CALL ..KDEL(MIN,ITIC,.TRUE.,.FALSE.)
      RETURN
      END
```

```
C
C
      SUBROUTINE SETLB(LB,KW,KVAR,IERR)
C
C  ROUTINE SETS LOAD BANK SWITCHES BY CONTROLLING A GROUP OF CONTACTS
C  IN I/OIS SYSTEM.
C
C  LB=1,2,OR 3 (DESIGNATES LOAD BANK NUMBER)
C  KW IS INTEGER ARRAY WITH RESISTIVE LOAD VALUES FOR PHASE 1,2, AND 3.
C  MAXIMUM OF 40 KW PER LOAD BANK.
C  KVAR IS ARRAY WITH REACTIVE LOAD VALUES.  MAXIMUM OF 30 KVAR PER
C  LOAD BANK.
C  IERR=0 IF NO ERRORS IN INPUT PARAMETERS; =1 IF ERRORS.
C
      INTEGER KW(3),KVAR(3),FACTOR,S12(3),S21(3),S20(3),S13(3)
      INTEGER OUTCOD(3),SCAN(3),TAB21(8),TAB12(6),TAB13(6),TAB20(6)
C
C  TAB12 THROUGH TAB20 ARE ARRAYS OF BIT CONFIGURATIONS WHICH WHEN
C  OUTPUT TO A GROUP OF CONTACTS OF THE I/OIS, SET UP THE SAME LOADS
C  AS THE SWITCH SETTINGS AVAILABLE ON THE LOAD BANK CONTROL CONSOLE.
C  THE NUMBERS 12,21,13,20 CORRESPOND TO THE SWITCH MARKINGS FOR
C  PHASE #1 OF LOAD BANK #1 USED IN SCHEMATIC DRAWING OF CONTROL
C  CONSOLE.
C
C  THIS ROUTINE, THEN, COMBINES THE REQUIRED BIT PATTERN TO ACHIEVE
C  THE LOAD SETTING REQUESTED AND OUTPUTS THIS CONFIGURATION TO THE
C  GROUP OF CONTACTS WHICH CONTROLS THE LOAD BANK BEING USED.
C
C
      DATA TAB12/0,Z8000,Z4000,ZC000,Z6000,ZE000/
      DATA TAB21/0,Z1000,Z0800,Z1800,Z0C00,Z1C00,Z0E00,Z1E00/
      DATA TAB13/0,Z0100,Z0080,Z0180,Z00C0,Z01C0/
      DATA TAB20/0,Z0020,Z0010,Z0030,Z0018,Z0038/
C  INITIALIZE ERROR FLAG
      IERR=0
C  CHECK FOR VALID LOAD BANK #
      IF(LB.LT.1.OR.LB.GT.3)GO TO 1000
C  FACTOR ACCOUNTS FOR PARALLELED LOAD BANKS
      FACTOR=2
      IF(LB.EQ.3)FACTOR=1
C  CHECK VALIDITY OF KW AND KVAR SETTINGS
      DO 10 I=1,3
      IF(KW(I).LT.0.OR.KW(I)/FACTOR.GT.40)GO TO 1000
10    IF(KVAR(I).LT.0.OR.KVAR(I)/FACTOR.GT.30)GO TO 1000
C  COMPUTE SWITCH SETTINGS
      DO 20 I=1,3
C  FOR HIGHEST KW SETTING, SELECT SWITCH CONFIGURATION
      IF(KW(I)/FACTOR.NE.40)GO TO 11
      S12(I)=6
      S21(I)=8
      GO TO 12
C  SELECT LESSER KW SWITCH SETTING
11    S12(I)=MOD(KW(I)/FACTOR,5) + 1
C  SELECT GREATER KW SETTING
      S21(I)=KW(I)/FACTOR/5 + 1
C  FOR HIGHEST KVAR SETTING, SELECT SWITCH SETTING
```

```
12    IF(KVAR(I)/FACTOR.NE.30)GO TO 13
      S13(I)=6
      S20(I)=6
      GO TO 20
C  SELECT LESSER KVAR SETTING
13    S13(I)=MOD(KVAR(I)/FACTOR,5) + 1
C  SELECT GREATER KVAR SETTING
      S20(I)=KVAR(I)/FACTOR/5 + 1
20    CONTINUE
C  SET UP WORDS TO BE OUTPUT TO I/OIS
      DO 30 I=1,3
30    OUTCOD(I)=TAB12(S12(I))+TAB13(S13(I))+TAB20(S20(I))+
     1TAB21(S21(I))
C  SET UP SCAN TABLE TO SERVE AS ADDRESS CONTROL FOR I/OIS
      DO 40 I=1,3
40    SCAN(I)=4Z4010+(LB-1)*3+I-1
C  OUTPUT CONTROL WORDS TO CONTACTS THROUGH I/OIS
      INLINE
        LDI,2  UFT
        REX,1
        DFC  OUTCOD,6      OUTPUT 3 WORDS
        BRU  OUT
        DFC  SCAN,3
UFT     DFC 0,3D00,#8400,0,0,0
OUT     NOP
      FINI
      RETURN
C  PRINT ERROR MESSAGE
1000  IERR=1
      WRITE(6,1)
1     FORMAT(' ***ERROR - SUBROUTINE SETLB',//,
     1' ILLEGAL SETTING REQUESTED')
      RETURN
      END
C
C
      SUBROUTINE SETRC(COM,IERR)
C
C  ROUTINE SETS CONTACTS IN I/OIS SYSTEM TO DESIRED STATE (1 OF 0)
C  COM(1) HAS NUMBER OF CONTACT, COM(2) HAS DESIRED STATE
C  IERR=0 IF NO ERRORS IN INPUT PARAMETERS
C
      INTEGER COM(1),WORDOUT,SCAN
      DATA WORDOUT/0/,SCAN/4Z4019/
C  INITIALIZE ERROR FLAG
      IERR=0
C  CHECK VALIDITY OF PARAMETERS
      IF(COM(1).LT.1.OR.COM(1).GT.13.OR.COM(2).LT.0.OR.COM(2).GT.1)
     1GO TO 1000
C  CONTACT NUMBERING AND ACTUAL ADDRESS ARE NOT EQUIVALENT.
C  CONTACTS NUMBERED 1-8 ARE CONTROLLED BY BITS 6-13 OF CONTROL
C  WORD.  CONTACTS 9-12 ARE CONTROLLED BY BITS 1-4.
C  THEREFORE, A SEPARATE ROUTINE IS REQUIRED TO DEVELOP THE CONTROL
C  WORD FOR EACH GROUP OF CONTACTS.
C
C  SELECT APPROPRIATE ROUTINE BASED ON CONTACT NUMBER
```

```
         IF(COM(1).GT.8)GO TO 2
C   ROUTINE FOR CONTACTS 1-8
C   REQUEST TO OPEN CONTACT?
         IF(COM(2).EQ.0)GO TO 1
C   SET BIT IN CONTROL WORD TO CLOSE CONTACT
         CALL SETBF(WORDOUT,COM(1)+5)
         GO TO 10
C   SET BIT IN CONTROL WORD TO OPEN CONTACT
1        CALL SETBT(WORDOUT,COM(1)+5)
         GO TO 10
C   ROUTINE FOR CONTACTS 9-12
C   REQUEST TO OPEN CONTACT?
2        IF(COM(2).EQ.0)GO TO 5
C   SET BIT TO CLOSE CONTACT
         CALL SETBF(WORDOUT,COM(1)-8)
         GO TO 10
C   SET BIT TO OPEN CONTACT
5        CALL SETBT(WORDOUT,COM(1)-8)
10       CONTINUE
C   OUTPUT CONTROL WORD TO I/OIS DEVICE
         INLINE
           LDI,2  UFT
           REX,1
           DFC WORDOUT,2
           BRU OUT
           DFC  SCAN,1
UFT        DFC 0,0000,#8400,0,0,0
OUT        NOP
         FINI
         RETURN
C   SET ERROR FLAG
1000     IERR=1
C   PRINT MESSAGE AND RETURN
         WRITE(6,100)
100      FORMAT(' ***ERROR - SUBROUTINE SETRC')
         RETURN
         END
C
C
         SUBROUTINE ACQUIRE(TIM)
C
C   ROUTINE PREPARES DATA STRUCTURES REQUIRED BY DATA ACQUISITION
C   TASK, DAC, MAKES POINTERS TO THESE STRUCTURES AVAILABLE TO
C   DAC, AND THEN ACTIVATES THE DAC TASK.
C
         INTEGER UFT(10,9),BUF(2946,9),IARRAY(30),CNT
         DATA MAXBUF/9/,CNT/2944/
C   INITIALIZE ERROR COUNT FOR PARITY ERRORS DISCOVERED BY
C   ROUTINE GETDATA
1        IERR=0
C   CALL ROUTINE WHICH INPUTS A BLOCK OF DATA FROM THE DATA
C   ACQUISITION SYSTEM VIA THE 4805 DEVICE
         CALL GETDATA(IARRAY,30,10,NV,IERR)
C   CHECK FOR PARITY OR CHECKSUM ERRORS
         IF(IERR.EQ.0)GO TO 2
C   PRINT MESSAGE TO USER AND LOOP UNTIL ERROR-FREE BLOCK IS INPUT
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  09:30:20        PAGE   10

           WRITE(6,3)IERR
     3     FORMAT(' ***4805 HAD',I5,' ERRORS - SUBROUTINE ACQUIRE',
          1//,' PARITY OR CHECKSUM ERRORS')
           GO TO 1
     C  DETERMINE TIME OF ONE BLOCK OF DATA
     2     TIME=(IARRAY(10)-IARRAY(2))*0.2E-6
     C  DETERMINE NUMBER OF DISK TRACKS NEEDED TO STORE DATA FOR
     C  REQUESTED TIME (TIM) AT PRESENT SAMPLING RATE
     C  NOTE:  MAXTRK = (1 BLOCK/TIME SEC) * (10 WORDS/BLOCK) *
     C                   (1 TRACK/23 SECTORS USED) * (1 SECTOR/128 WORDS)
           MAXTRK=(10.*TIM)/(TIME*2944.)+1
     C
     C  CREATE UFT'S FOR WRITING TO DISK AND BUFFER AREAS FOR READING
     C  FROM 4805 USING DATA CHAINING
     C  NOTE:  DATA INPUT SCHEME USES CIRCULAR FILLING OF 9 BUFFERS
     C  AND STORAGE OF EACH BUFFER ON A SEPARATE DISK FILE.
           DO 10 I=1,MAXBUF
     C  SET WORD COUNT FOR DATA CHAINING
           BUF(CNT+1,I)=IAND(-CNT,#7FFF)
     C  SET UP UFT'S
           UFT(1,I)=0
           UFT(2,I)=ICAN(30)
           UFT(3,I)=#2A400
           DO 10 J=4,10
     10    UFT(J,I)=0
     C  SET POINTERS TO NEXT BUFFER FOR DATA CHAINING
           M=MAXBUF-1
           DO 20 I=1,M
     20    BUF(CNT+2,I)=IADR(BUF(1,I+1))
     C  SET POINTER IN LAST BUFFER TO POINT BACK TO FIRST BUFFER
           BUF(CNT+2,MAXBUF)=IADR(BUF(1,1))
     C
     C  PUT MAXTRK ON DISK, SO THAT ANALYSIS ROUTINES WILL KNOW EXTENT
     C  OF STORED DATA
     C
     C  SET RANDOM ADDRESS POINTER IN UFT
           UFT(4,1)=23
     C  READ FROM DISK TO POSITION PAST CALIBRATION DATA
     C  STORED BY ROUTINE CALAD
           CALL READ(BUF(1,1),128,UFT(1,1),.TRUE.)
     C  RESET RANDOM ADDRESS POINTER IN UFT
           UFT(4,1)=23
           BUF(1,1)=MAXTRK
           CALL WRITE(BUF(1,1),128,UFT(1,1),.TRUE.)
     C
     C  ASSEMBLY LANGUAGE ROUTINE TO LOAD POINTER IN TASK COMMUNICATION
     C  AREA WITH ADDRESS OF UFT'S AND DATA BUFFERS AND NUMBER OF TRACKS
     C  TO BE ACQUIRED.
     C  THESE PARAMETERS WILL BE PICKED UP BY THE DAC TASK WHICH WILL
     C  BE TAKEN OUT OF A WAIT STATE AT THE END OF THIS ROUTINE.
     C
           INLINE
             LDI,2  @BFR       POINTER NAME IS BFR
             LDI,3  BUF
             REX,#4E           STORE ADDRESS OF BUFFER IN POINTER
             LDI,2  @TRK       FLAG NAME
```

```
          LDM,3  MAXTRK
          REX,#4E        STORE NUMBER OF TRACKS
          LDI,2  @UFT     POINTER NAME
          LDI,3  UFT
          REX,#4E       STORE POINTER
          REX,#15
          DFC  0
          DFC  @DAC     RESUME DAC TASK
        FINI
        RETURN
        END
 C
 C
 C   THE FOLLOWING ROUTINES ARE USED TO INPUT FREE-FORMAT REAL
 C   AND INTEGER PARAMETERS FOR GENERATOR TEST SEQUENCE MAINTENANCE
 C   COMMANDS.
 C
 C
        SUBROUTINE READI(N,IARRAY)
 C
 C   ROUTINE INPUTS ASCII STRING IN FREE FORMAT AND RETURNS N
 C   EQUIVALENT INTEGER VALUES IN IARRAY
 C
        INTEGER IARRAY(1),BUF(74),BUFSIZ
        DATA BUFSIZ/74/
 C   INPUT STRING OF ASCII CHARACTERS
 100    READ(7,1000)BUF
 1000   FORMAT(74A1)
 C   CALL ROUTINE TO FORMAT AND CONVERT TO INTEGER
        CALL INTFORM(N,BUF,BUFSIZE,IARRAY,IERR)
 C   CHECK FOR ERROR IN PARAMETERS
        IF(IERR.EQ.0)RETURN
 C   ECHO INVALID CHARACTER TO USER
        WRITE(8,2000)BUF(IERR)
 2000   FORMAT(' INVALID CHARACTER -',A1,5X,'RE-ENTER LINE')
 C   GET NEW STRING
        GO TO 100
        END
 C
 C
        SUBROUTINE INTFORM(NPAR,STRING,LTH,INT,IERR)
 C
 C   ROUTINE CONVERTS ASCII STRING (STRING) OF LENGTH, LTH, TO
 C   NPAR EQUIVALENT INTEGERS STORED IN ARRAY INT.  IERR IS SET
 C   IF ANY CHARACTER IN STRING IS NOT A VALID DIGIT.
 C
        INTEGER INT(1),STRING(1)
 C   INITIALIZE ERROR FLAG
        IERR=0
 C   INITIALIZE POINTER INTO STRING
        IPNT=1
 C   CONVERT EACH INTEGER
        DO 100 I=1,NPAR
 C   INITIALIZE VALUE
        INT(I)=0
 C   CHECK IF POINTER EXCEEDS STRING LENGTH
```

```
*D MODCOMP SOURCE EDITOR      DATE  12/27/79  09:30:20         PAGE   12

 10      IF(IPNT.GT.LTH)GO TO 100
 C   CHECK FOR DELIMITER
 20      IF(STRING(IPNT).NE.' '.AND.STRING(IPNT).NE.',')GO TO 30
 C   INCREMENT POINTER
         IPNT=IPNT+1
 C   GET NEXT CHARACTER
         GO TO 10
 C   CHECK FOR VALID DIGIT
 30      IF(STRING(IPNT).LE.'9'.AND.STRING(IPNT).GE.'0')GO TO 40
 C   SET ERROR FLAG AND RETURN
         IERR=IPNT
         RETURN
 C   CONVERT TO INTEGER AND ADD TO ACCUMULATION OF VALUE
 40      INT(I)=INT(I)*10 + (STRING(IPNT)-'0')/256
 C   INCREMENT POINTER
         IPNT=IPNT+1
 C   CHECK FOR END OF STRING OR DELIMITER
         IF(IPNT.LE.LTH.AND.STRING(IPNT).NE.' '.AND.
        1STRING(IPNT).NE.',')GO TO 30
 100     CONTINUE
         RETURN
         END
```

APPENDIX C
DSC

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  16:24:56        PAGE   1

          PROGRAM DSC
      C
      C   FOLLOWING ROUTINE PROCESSES DRIVE STAND COMMAND
      C   INSTRUCTIONS RECEIVED FROM ROUTINE RUNSYS.
      C   RESULTS FROM THIS ROUTINE ARE OUTPUT (VIA DEVICE OUT=A01)
      C   TO THE 8080 DRIVE STAND CONTROLLER WHICH PROVIDES
      C   ACTUAL CONTROL OF THE DRIVE STANDS.
      C   THIS ROUTINE HAS BEEN CATALOGUED IN THE FORM OF AN
      C   AUTO-START TASK SO THAT IT EXECUTES CONTINUOUSLY UNLESS
      C   FORCED INTO A HOLD STATE.
      C
      C   * * * * * * * * * * * * * * * * * * * * * * *
      C   * * *  CATALOGUED ON LM  9,26,79  * * * * * * * * * * * *
      C   * * * * * * * * * * * * * * * * * * * * * * *
      C
      C   EXTERNAL SUBROUTINES REQUIRED:
      C
      C    NAME          LOCATION
      C
      C   1. SAFCHECK       UL
      C   2. WRITE          LB
      C   3. READ           LB
      C   4. WAIT           UL
      C   5. TERMIN         LB
      C   6. HOLD           LB
      C
      C
          INTEGER LASTCOM(3),BUF(4),TEMP(2),IUFT(10),OUFT(10)
          INTEGER ERRCNT
          REAL DSCALE(3)
          LOGICAL TESTB
          DATA IUFT/0,Z3A70,ZE200,0,1,5*0/
          DATA OUFT/0,Z611C,ZE200,0,8,5*0/
      C   NOTE:  #3A70=@IN; #611C=@OUT
      C
      C   INITIALIZE COMMAND; THIS COMMAND WILL BE PROCESSED UNTIL
      C   A VALID COMMAND IS RECEIVED FROM RUNSYS.
      C   NOTE:  LASTCOM(1)=DS#,LASTCOM(2)=SPEED,LASTCOM(3)=RATE
          LASTCOM(1)=1
          LASTCOM(2)=0
          LASTCOM(3)=0
      C   FOLLOWING IS AN ASSEMBLY LANGUAGE ROUTINE WHICH SETS FLAGS
      C   IN A TASK COMMUNICATION AREA SO THAT LATER PART OF THIS
      C   ROUTINE DOES NOT TRY TO PROCESS A NEW COMMAND UNTIL ONE IS
      C   AVAILABLE.
      C   TASK COMMUNICATION AREA IS HANDLED BY CUSTOM REX CALLS #4F
      C   AND #4D.  SEE LISTING OF O/S FOR FULL EXPLANATION OF THESE
      C   REX CALLS
      C
          INLINE
            LDI,2 @DSC      FLAG IS CALLED DSC
            ZRR,3
            REX,#4E     LOAD FLAG WITH ZERO
      *   MUST ALSO ZERO SBF FLAG, WHICH UNTIL CLEARED BY RUNSYS
      *   WILL CAUSE SAFETY CHECKING TO BE IGNORED.
            LDI,2  @SBF
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  16:24:56        PAGE   2

              ZRR,3
              REX,#4E
           FINI
      100  CONTINUE
      C  PERFORM CHECK OF SAFETY PARAMETERS
           CALL SAFCHECK
      C  TEST DSC FLAG.  IF ZERO, KEEP PROCESSING CURRENT COMMAND.
      C  IF NONZERO, DSC FLAG POINTS TO NEW COMMAND.  LOAD THIS
      C  NEW COMMAND INTO ARRAY, LASTCOM.
           INLINE
              LDI,2 @DSC
              REX,#4D      GET FLAG INTO REG. 3
              TRRB,3,3 $+4      CHECK FOR ZERO
              BRU A001      IF ZERO, TAKE THIS BRANCH
      *  IF FLAG NOT ZERO, JUMP IS MADE TO FOLLOWING INSTRUCTIONS
              LDX,2,3      REG 2 POINTS TO START OF NEW COMMAND
              STM,2 LASTCOM      GET FIRST WORD
              ABR,3,15
              LDX,2,3
              STM,2 LASTCOM+1      GET SECOND WORD
              ABR,3,15
              LDX,2,3
              STM,2 LASTCOM+2      GET THIRD WORD
      *  ZERO DSC FLAG, SO THAT UNTIL RUNSYS ISSUES A NEW COMMAND
      *  (AND CHANGES FLAG) COMMAND IN LASTCOM WILL BE PROCESSED.
              LDI,2 @DSC
              ZRR,3
              REX,#4E
      A001    NOP
           FINI
      C  CHECK VALIDITY OF DRIVE STAND COMMAND
           IF(LASTCOM(1).LT.1.OR.LASTCOM(1).GT.3.OR.
          1   LASTCOM(2).LT.0.OR.LASTCOM(2).GT.10000.OR.
          2   LASTCOM(3).LT.0.OR.LASTCOM(3).GT.10)GO TO 400
      C  SPEED AND RATE COMMANDS MUST BE CONVERTED TO FORMAT
      C  REQUIRED BY 8080 CONTROLLER.
      C  I.E., 12 BITS OF 1 (4095) = MAX. OUTPUT
           DO 302 I=1,2
      302  TEMP(I)=IFIX(FLOAT(LASTCOM(I+1))*0.4095)
      C  ARRAY BUF WILL CONTAIN COMMAND STRING SENT TO 8080 DS
      C  CONTROLLER.  THE FOLLOWING INSTRUCTIONS PACK ARRAY BUF.
      C
      C  LOAD BUF(1) WITH 0D=CR WHICH SIGNALS BEGINNING OF
      C  COMMAND AND DS# IN ASCII.
           BUF(1)=IOR(4Z0D30,LASTCOM(1))
      C  THE FOLLOWING INSTRUCTIONS USE FUNCTION IASCII TO CONVERT
      C  THE SPEED AND RATE COMMANDS TO ASCII AND POSITION THEM
      C  IN THE PROPER BYTE POSITION OF THE WORD.
      C  I.E., SPEED = MSB(BUF(2)),LSB(BUF(2)),MSB(BUF(3))
      C        RATE = LSB(BUF(3)),MSB(BUF(4)),LSB(BUF(4))
           BUF(2)=IOR(IASCII(TEMP(1),3,1),IASCII(TEMP(1),2,0))
           BUF(3)=IOR(IASCII(TEMP(1),1,1),IASCII(TEMP(2),3,0))
           BUF(4)=IOR(IASCII(TEMP(2),2,1),IASCII(TEMP(2),1,0))
      C  OUTPUT COMMAND STRING TO 8080 DS CONTROLLER.  WAIT UNTIL
      C  OUTPUT OPERATION IS COMPLETE BEFORE CONTINUING.
           CALL WRITE(BUF,8,OUFT,.TRUE.)
```

```
C  INPUT ONE BYTE FROM 8080 (VIA DEVICE IN=AI1).  THIS
C  IS THE DS# WHICH THE CONTROLLER ECHOES BACK TO INDICATE
C  THAT IT IS OPERATING CORRECTLY.  THIS INPUT OPERATION IS
C  DONE IN QUICK-RETURN MODE, SO THAT CONTROL IS RETURNED
C  TO THE ROUTINE AFTER THE OPERATION IS INITIATED.
       CALL READ(BUF,1,IUFT,.FALSE.)
C  DELAY 20 TICKS TO ALLOW INPUT TO FINISH.
       CALL WAIT(20,0,DUMMY)
C  CHECK TO SEE IF INPUT HAS FINISHED (BIT 15 OF IUFT = 0)
C  NOTE:  TESTB RETURNS TRUE FOR 0.
       IF(TESTB(IUFT(1),15))GO TO 303
C  IF INPUT HAS NOT FINISHED, TERMINATE THE INPUT OPERATION.
       CALL TERMIN(IUFT)
C  INCREMENT ERROR COUNT WHICH KEEPS TRACK OF HOW MANY TIMES
C  8080 HAS FAILED TO ECHO
       ERRCNT=ERRCNT+1
C  ALLOW MAX. OF 2 ERRORS BEFORE HOLDING
       IF(ERRCNT.GT.2)GO TO 500
       GO TO 400
C  IF INPUT OPERATION WAS FINISHED, CHECK TO SEE IF DS#
C  ECHOED MATCHES DS# SENT
C  NOTE:  SINCE DS# WAS INPUT AS 1 BYTE, IT MUST BE SHIFTED
C  RIGHT BEFORE COMPARISON.  ALSO CONVERTED TO ASCII.
303    IF(ISHFT(BUF(1),-8)-2Z30.NE.LASTCOM(1))GO TO 500
C  IF DS#'S MATCH, RESET ERROR COUNT
       ERRCNT=0
400    CONTINUE
C  DELAY NEEDED HERE FOR PROPER TIMING SYNCHRONIZATION WITH
C  8080 DS CONTROLLER
       CALL WAIT(200,0,DUMMY)
C  LOOP BACK TO GET NEW COMMAND
       GO TO 100
500    CONTINUE
C  IF 8080 DOESN'T ECHO CORRECTLY, PUT TASK IN HOLD STATE
       CALL HOLD(.TRUE.)
C  AFTER TASK IS RESUMED, GO BACK TO LOOP
       GO TO 400
       END
C
C
       FUNCTION IASCII(ICMD,NHEX,NBYTE)
C
C  ICMD IS 12 BIT COMMAND TO DRIVE STAND.  THIS ROUTINE
C  EXAMINES ONE HEX DIGIT (4 BITS) OF ICMD (POINTED TO BY NHEX)
C  ,CONVERTS THIS HEX DIGIT TO ASCII, AND POSITIONS IT IN
C  THE UPPER OR LOWER BYTE (DETERMINED BY NBYTE) OF A WORD
C  WHICH IS OUTPUT TO THE 8080 DRIVE STAND CONTROLLER.
C
C  SHIFT HEX DIGIT OF INTEREST TO RIGHT-MOST POSITION IN WORD
C  AND MASK OFF OTHER DIGITS.
       KDIG=IAND(ISHFT(ICMD,(1-NHEX)*4),1ZF)
C  IF DIGIT > 9, ASCII CODE IS DIFFERENT
       IF(KDIG.GT.9)GO TO 1
C  CONVERT TO ASCII AND POSITION IN WORD
       IASCII=ISHFT(KDIG+2Z30,NBYTE*8)
       RETURN
```

```
 1      IASCII=ISHFT(KDIG+2Z37,NBYTE*8)
        RETURN
        END
```

APPENDIX D
DAC

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  16:25:13        PAGE   1

         PROGRAM DAC
C
C  THIS ROUTINE IS CATALOGUED AS A TASK.  IT CAUSES DATA TO BE
C  ACQUIRED FROM THE DATA ACQUISITION SYSTEM AND STORED ON DISK.
C  THIS DATA ACQUISITION IS PERFORMED ACCORDING TO PARAMETERS
C  SPECIFIED IN THE NEWSYS ROUTINE.
C
C  * * * * * * * * * * * * * * * * * * * * * * *
C  * * * * *  CATALOGUED ON LM  9,26,79  * * * * * * * *
C  * * * * * * * * * * * * * * * * * * * * * * *
C
C
         LOGICAL TESTB
         INTEGER BUF1(10),BUF2(14),TEMP2,TEMP1,TEMP3
         INTEGER FLAG,TRKCNT,ERROR
         DATA BUF1/' ERROR HAS  OCCURRED'/
         DATA BUF2/' POSSIBLE ERROR HAS OCCURRED'/
         DATA MAXBUF/9/
         DATA TEMP2,TEMP3/0,0/
         MAXINT=(MAXBUF*16)+428200
C  MAXINT REFLECTS MAXIMUM ALLOWABLE NUMBER OF INTERRUPTS
C
   99 FLAG=0
C  FLAG STORES INDICATION OF INTERRUPT.
C
C  FOLLOWING IS ASSEMBLY LANGUAGE ROUTINE THAT FIRST
C  PUTS TASK IN WAIT.  TASK IS REMOVED FROM WAIT BY
C  SUBROUTINE ACQUIRE WHICH HAS JUST FILLED PARAMETER
C  PASSING AREA.  TASK THEN PICKS UP PARAMETERS AND
C  READS DATA FROM 4805 INTO BUFFER AREA.
C
         INLINE
DEV    EQU              3     DEVICE #3 IS 4805
SECAMT EQU              23              ;23 SECTORS/TRACK USED
WDCNT EQU               SECAMT*128      ;WORDCOUNT PER TRACK
          LDI,2  CIRLOC
          STM,2  #80+DEV
          STM,2  #C0+DEV          SET UP INT. ROUTINES
          LDI,1  #4500
          OCA,1,DEV               TERMINATE 4805
WAIT      REX,#11           PUT TASK IN WAIT MODE
          LDI,2  @BFR
          REX,#4D
          TRRB,2,2  PEXIST        IF REG=/ 0; REQ. PAR. EXISTS
          BRU  WAIT               IF REG2 = 0; PAR. DOES NOT EXIST
PEXIST TRRB,3,3  PAROK  IF REG3=/0; PAR HAS BEEN LOADED
          BRU WAIT
PAROK     STM,3  BUF        GET ADDR. OF BUFFER
          LDI,2  @BFR
          LDI,3  0
          REX,#4E           LOAD ZERO IN BFR
          LDI,2  @TRK
          REX,#4D
          STM,3  MAXTRK     GET NUMBER OF TRACKS
          LDI,2  @UFT
          REX,#4D
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  16:25:13        PAGE   2

           STM,3  UFTT        GET ADDR. OF UFT
           LDM,3  BUF
        LDI,2          INT            ;USE INT ROUTINE "INT"
        STM,2          #80+DEV        ;SET UP DI INTERRUPT TRAP
        LDI,1          #8200          ;SET UP OBMM,0,0 INSTR
        STM,1          INT            ;PUT IN INT
        LDI,1          CIRLOC         ;GET ADDR OF DUMMY INT ROUTINE
        STM,1          #C0+DEV        ;SET UP SI LOCATION
        LDI,1          #4000
        OCA,1,DEV                     ;DISCONNECT SI & DI
        LDI,1          #00C1
        OCA,1,DEV                     ;RESET SI FLAGS AND SET UP COMMAND BFR.
        ISA,1,DEV               ;GET  STATUS
        TBRB,1,0       STOK           ;CHECK STATUS ERRORS
  STEP  NOP                           ;***FUTURE ERROR ROUTINE
  STOK  LDI,1     -WDCNT$#7FFF  ;WORD COUNT FOR TRACK WITH DATA CHAINING
        STM,1          #60+DEV        ;STORE IN DMP AREA
        LDM,2          BUF            ;GET ADDR OF FIRST BUFFER
        STM,2          #70+DEV        ;STORE IN DMP AREA
        LDI,1          #F300
        OCA,1,DEV      ;INITIATE INPUT
        BRU            DOMORE         ;GET OUT OF ASSEMBLY CODE
  *     INTERRUPT ROUTINE FOR 4805
  INT   OBMM,0         FLAG           ;SET BIT IN FLAG TO IND INT ENTERED
        ABMM,11        INT            ;SET UP FOR NEXT INTERRUPT TO SET NEXT BIT IN
        STM,1          RSAVE          ;SAVE REG1
        LDM,1          INT            ;GET OBMM INSTR IN REG1
        CRMB,1         MAXINT,$+4,INTOK ;CHECK FOR VALID OBMM INSTR
        LDI,1          #8200          ;SET UP OBMM,0,0 IN REG1
        STM,1          INT            ;PUT IN INT FOR NEXT TIME
  INTOK LDI,1          #7000          ;SET UP COMMAND WORD
        OCA,1,DEV                     ;RE-CONNECT INTERRUPTS
        LDM,1          RSAVE          ;RESTORE REG1
  CIRLOC CIR                          ;RETURN(ALSO DUMMY INT ROUTINE LOC.)
  RSAVE RES            1              ;SAVE AREA FOR REG1
  UFTT    RES    1
  BUF  RES     1
  DOMORE               NOP            ;END OF MACRO ASSEMBLER CODE
        FINI
        TRKCNT=0
  C
  C  CHECK FOR ERRORS IN READING DATA FROM 4805.
  C
        IPERR=0
  C  SET POSSIBLE ERROR COUNT TO 0
        ERROR=0
  C  SET ERROR COUNT TO 0
        I=0
  11    I=I+1
        IF(I.GT.MAXBUF)I=1
        IF(TESTB(FLAG,I))GO TO 11
        J=I
        IF(J.GE.MAXBUF)J=0
  C
  C  ASSEMBLY CODE TO ACCESS UFTT(1,J+1) NEEDED IN NEXT
  C  FORTRAN STATEMENT.
```

```
C
      INLINE
        LDI,3  10
      MPM,3         J              REG3=J*10
      ADM,3         UFTT           ;REG3=UFTT+ J*10
        LDX,3,3
        STM,3  TEMP1     TEMP1= UFTT(1,J+1)
      FINI
      IF(.NOT.TESTB(TEMP1,16))IPERR=IPERR+1
C
C  ASSEMBLY CODE TO ACCESS UFTT(1,I),BUF(1,I) AND
C  TO LOAD UFTT(4,I) WITH TRKCNT*24.
C
      INLINE
        LDM,3  I
        SBR,3,15          REG3= I-1
        LDI,2  10
        MPR,3,2           REG3= (I-1)*10
      ADM,3         UFTT            ;REG3=UFTT+(I-1)*10
        STM,3  TEMP3
        LDX,4,3
        STM,4  TEMP1      TEMP1= UFTT(1,I)
        ABR,3,15
        ABR,3,14          REG3 POINTS TO UFTT(4,I)
        LDI,5  24
      MPM,5         TRKCNT         ;REG5=TRKCNT*24
        STX,5,3           UFTT(4,I)= TRKCNT*24
        LDM,2  I
        SBR,2,15          REG2 = I-1
        LDI,3  2946
      MPR,3,2                      ;REG3=(I-1)*2946
      ADM,3         BUF            ;REG3=BUF+(I-1)*2946
        STM,3  A001
        LDX,3,3
        STM,3  TEMP2      TEMP2= BUF(1,I)
      FINI
15    IF(TESTB(TEMP1,16))GO TO 12
      ERROR=ERROR+1
13    IF(.NOT.TESTB(TEMP1,16))GO TO 13
12    CALL SETBT(FLAG,I)
      INLINE
        LDI,2  WDCNT*2
        STM,2  A001+1
        LDM,2  TEMP3
        REX,#81
A001  DFC  0,0
        REX,#14
        DFC  #8000,4      DELAY TO ALLOW OTHER TASKS TO RUN
      FINI
      TRKCNT=TRKCNT+1
      IF(TRKCNT.GE.MAXTRK)GO TO 100
      GO TO 11
100   CONTINUE
      INLINE
      LDI,1         #4500
      OCA,1,DEV                    ;TERMINATE 4805 DEVICE
```

```
          FINI
          IF(ERROR.NE.0)CALL MESSAG(MES1,.TRUE.)
          IF(IPERR.NE.0)CALL MESSAG(MES2,.TRUE.)
          GO TO 99
    999   STOP
          END
```

APPENDIX E

DRIVE STAND CONTROLLER

The drive stand controller is a microprocessor based system which provides control of the test facility drive stands. Actual control of a drive stand is achieved by supplying a DC voltage to the drive stand power supply. A voltage of 0-10 VDC input to the power supply produces a drive stand speed of 0-10,000 rpm. The drive stand controller receives control commands from the minicomputer (via the procedure explained in the main text of this manual), processes these commands, and issues the appropriate DC voltage signals to the drive stand power supply.

The drive stand controller consists of an Intel 8080 microprocessor, associated peripheral components, and a digital-analog conversion (DAC) circuit. Following is a discussion of how these components interact to process drive stand commands.

The microprocessor basically inputs commands from the minicomputer, converts the commands to the format required by the DAC circuit, and outputs the processed commands to the DAC at timed intervals. The DAC converts the digital signal to the appropriate DC voltage.

The microprocessor contains a read-only memory (ROM) programmed with a control routine. This control routine directs the system components to perform the input, process, output sequence. The source language for the control routine is Intel PLM. A copy of this code is attached.

Commands are input from the minicomputer via a peripheral interface device (8255). The 8255 receives a serial command from the minicomputer, converts this command to parallel, and issues an interrupt to the microprocessor CPU signalling that an input command is available.

The CPU accepts the command and converts it to the format required by the DAC to produce the DC voltage which will achieve the requested drive stand speed. To control acceleration rate, this DC voltage will be output in timed steps.

An interval timer (8253) outputs a command from the microprocessor to the DAC every 10 milliseconds. Therefore, for example, to accelerate the drive stand from 0 rpm to 1,000 rpm at 10,000 rpm/sec (maximum acceleration rate), the following series of DC voltage steps would be output from the DAC.

DC voltage	time (msec)
0	0
0.1	10
0.2	20
0.3	30
0.4	40
0.5	50
0.6	60
0.7	70
0.8	80
0.9	90
1.0	100

The minicomputer and the drive stand controller operate in a handshaking mode as follows. A time count is decremented by the control routine on each iteration. Input of a new command from the minicomputer resets this count. Thus to maintain a speed, the minicomputer must periodically output the command which defines that speed. Details of how this is achieved are included in the main text of this manual.

If the controller count reaches zero, which means the minicomputer has failed to issue a new command, the drive stand controller is programmed to force the drive stands to zero speed. In addition, the controller must echo the DS# portion of the drive stand command to the minicomputer. If the minicomputer does not receive this echo, it puts itself in a hold state. This forces the drive stand controller to take the shutdown action described above, since no further commands will be issued by the minicomputer. Thus if communication is broken between the minicomputer and the drive stand controller, the drive stands are brought to zero speed.

Figures E-1 and E-2 are circuit drawings of the drive stand controller. Figure E-1 contains the additional circuitry added to a basic Intel MCS-80 microprocessor kit to achieve the structure needed for the drive stand controller. For further details on the MCS-80 kit refer to Intel manual "MCS-80 System Design Kit User's Guide." Figure E-2 is the digital-to-analog conversion circuit which produces the analog signal output to the drive stand power supply.

*D MODCOMP SOURCE EDITOR   DATE 01/03/80 14:44:20   PAGE   1

```
/*  FOLLOWING IS THE PLM SOURCE CODE FOR THE CONTROL  */
/*  ROUTINE STORED IN THE EPROM OF THE INTEL 8080 MICRO-  */
/*  PROCESSOR OF THE DRIVE STAND CONTROLLER  */
     DECLARE FLAG BYTE;
     DECLARE (SPEED,RATE)(4)BYTE;
     DECLARE (DS,K,TIME)BYTE;
     DECLARE (SPD,RT,CONTROL)(4)ADDRESS;
     DECLARE BUF(7) BYTE;
READMC% PROCEDURE INTERRUPT 1;
/*  ROUTINE TO READ COMMAND FROM MODCOMP  */
/*  INVOKED BY TXRDY OF 8251 AT LEVEL 1  */
     DECLARE (VALUE,POINT) BYTE;
/*  INPUT COMMAND AND STRIP OFF PARITY  */
     VALUE = INPUT(0FAH) AND 7FH;
/*  CHECK POINTER INTO BUFFER  */
     IF ((POINT<7) OR (VALUE=0DH))
       THEN DO;
              IF(VALUE=0DH) /* 0D=CR;BEGINNING OF NEW COMMAND */
                THEN POINT=0; /* RESET POINTER */
                ELSE DO;
/*  CHANGE VALUE TO ASCII AND STORE IN COMMAND BUFFER  */
                       IF(VALUE< 3AH)
                         THEN BUF(POINT) = VALUE - 30H;
                         ELSE BUF(POINT) = VALUE - 37H;
/*  CHECK FOR VALID HEX DIGIT  */
                      IF(BUF(POINT) < 10H)
                        THEN DO;
                               POINT = POINT + 1; /* INCREMENT POINTER */
/*  IF POINTER EXCEEDS 6, SET FLAG TO INDICATE COMMAND BUFFER IS FULL */
                               IF (POINT=7)
                                 THEN FLAG=0;
                              END;
                     END;
          END;
/*  RESET INTERRUPT CONTROLLER  */
          OUTPUT(0F6H) = 20H;
END READMC;
/*                                                              */
/*                                                              */
ZEROSPEEDS% PROCEDURE;
/*  ROUTINE WHICH FORCES ALL SPEED AND RATE OUTPUT COMMANDS  */
/*  TO ZERO.  ROUTINE IS CALLED AT INITIALIZATION AND WHENEVER */
/*  MODCOMP FAILS TO ISSUE A NEW COMMAND BEFORE TIMER EXPIRES  */
    DO K= 1 TO 3;
       SPD(K)=0;
       RT(K)=0;
    END;
    RETURN;
END ZEROSPEEDS;
/*                                                              */
/*                                                              */
DRIVESTAND% PROCEDURE INTERRUPT 2;
/*  ROUTINE TO UPDATE DRIVESTAND SPEEDS  */
/*  INVOKED BY 10 MSEC TIMER AT LEVEL 2  */
     DECLARE (HBB,LBB,K) BYTE;
/*  ENABLE OTHER INTERRUPTS, SO THAT THIS ROUTINE CAN BE  */
```

```
*D MODCOMP SOURCE EDITOR      DATE  01/03/80  14:44:20          PAGE   2

 /*   INTERRUPTED BY LEVEL 1 (INPUT)  */
      ENABLE;
      TIME=TIME-1;  /*  DECREMENT COMMAND TIMER  */
 /*   CHECK TO SEE IF TIMER HAS EXPIRED  */
      IF(TIME=0)
         THEN CALL ZEROSPEEDS; /* IF SO, FORCE DRIVES TO ZERO */
 /*   OTHERWISE, CALCULATE NEW OUTPUT COMMAND TO BE ISSUED TO D/A */
         ELSE DO K=1 TO 3;
 /*   IF REQUESTED SPEED IS GREATER THAN LAST SPEED COMMAND  */
 /*   ISSUED PLUS ONE INCREMENT OF RATE REQUESTED  */
            IF((SPD(K)>(CONTROL(K)+RT(K)) AND(RT(K)<>0))
 /*   THEN NEW SPEED COMMAND IS SUM OF LAST COMMAND PLUS ONE  */
 /*   INCREMENT OF RATE  */
               THEN CONTROL(K)=CONTROL(K)-RT(K);
 /*   ELSE, CHECK IF REQUEST IS FOR DECELERATION  */
               ELSE IF ((SPD(K)<(CONTROL(K)-RT(K)))AND(RT(K)<>0)
                 AND (CONTROL(K) > RT(K)))
 /*   IF VALID DECELERATION, NEW COMMAND IS LAST COMMAND MINUS  */
 /*   RATE  */
                     THEN CONTROL(K)=CONTROL(K)-RT(K);
 /*   OTHERWISE, NEW COMMAND IS SPEED REQUESTED  */
                     ELSE CONTROL(K)=SPD(K);
 /*   NOTE:  SINCE THIS ROUTINE EXECUTES EVERY 10 MSEC, A RATE OF  */
 /*      1 CORRESPONDS TO 100 RPM/SEC  */
 /*                                                                */
 /*   BITS MUST BE INVERTED BEFORE OUTPUTTING TO D/A CONVERTER  */
            H8B=NOT(HIGH(CONTROL(K)) + (TIME AND 0F0H));
 /*   NOTE:  UPPER HEX DIGIT OF TIMER IS OUTPUT TO A 7 SEGMENT  */
 /*      LED DISPLAY ON CONTROLLER  */
            L8B=NOT(LOW(CONTROL(K)));
 /*   DISABLE INTERRUPTS, SO THAT FOLLOWING OUTPUT ROUTINE IS  */
 /*   NOT INTERRUPTED  */
            DISABLE;
 /*   OUTPUT COMMAND TO PAIR OF OUTPUT PORTS DETERMINED BY K  */
            DO CASE K-1;
 /*   DRIVE STAND #1 OUTPUT  */
              DO;
                OUTPUT(17H)=L8B;
                OUTPUT(37H)=H8B;
              END;
 /*   DRIVE STAND #2 OUTPUT  */
              DO;
                OUTPUT(57H)=L8B;
                OUTPUT(77H)=H8B;
              END;
 /*   DRIVE STAND #3 OUTPUT  */
              DO;
                OUTPUT(97H)=L8B;
                OUTPUT(0B7H)=H8B;
              END;
            END;
 /*   RE-ENABLE INTERRUPTS  */
         ENABLE;
         END;
 /*   RESET INTERRUPT CONTROLLER  */
      OUTPUT(0F6H)=20H;
```

```
MD MODCOMP SOURCE EDITOR      DATE  01/03/80  14:44:22          PAGE   3

            RETURN;
        END DRIVESTAND;
        /*                                                              */
        /*                                                              */
        ECHO% PROCEDURE(CHAR);
        /*  ROUTINE TO ECHO CHARACTER TO MODCOMP  */
            DECLARE CHAR BYTE;
        /*  WAIT UNTIL TXRDY IS 1; I.E., TRANSMITTER IS READY  */
            DO WHILE (INPUT(0FBH)AND 01)=0;
            END;
        /*  CHANGE CHARACTER TO ASCII AND OUTPUT TO MODCOMP  */
            OUTPUT(0FAH)=CHAR+30H;
        /*  SAVE THIS CHARACTER (WHICH IS DS#) IN VARIABLE K BY INPUTTING */
            K=INPUT(0FAH);
            RETURN;
        END ECHO;
        /*                                                              */
        /*                                                              */
        STARTCOUNT% PROCEDURE;
        /*  ROUTINE TO INITIALIZE DEVICES  */
        /*  INITIALIZE COUNTER, 8253  */
            OUTPUT(0D7H)=036H;  /* MODE 3, BINARY COUNT  */
            OUTPUT(0D6H)=30H;  /*  LSB OF COUNT  */
            OUTPUT(0D6H)=0;  /*  MSB OF COUNT  */
        /*  NOTE:  0030H GIVES A COUNT OF 10 MSEC  */
        /*  INITIALIZE INTERRUPT CONTROLLER, 8259  */
            OUTPUT(0F6H)=12H;  /* ICW1 - USE NORMAL INTERRUPT TRAPS  */
            OUTPUT(0F7H)=0;  /* ICW2 - COMPLETES ICW1  */
            OUTPUT(0F7H)=0F9H;  /* OCW1 - MASK ALL INTERRUPTS EXCEPT 1&2 */
        /*  INITIALIZE RS232 PORT, 8251  */
            OUTPUT(0FBH)=0CFH;  /* 2 STOP BITS, NO PARITY, 8 BIT CHAR */
                                /*        64 X BAUD RATE              */
            OUTPUT(0FBH)=27H;  /* ENABLE DEVICE  */
             RETURN;
        END STARTCOUNT;
        /*                                                              */
        /*                                                              */
        /*  MAIN ROUTINE WHICH ACCUMULATES COMMANDS INPUT BY READMC  */
        /*  AND MAKES THEM AVAILABLE TO DRIVESTAND  */
            CALL STARTCOUNT;  /*  INITIALIZE DEVICES  */
            CALL ZEROSPEEDS;  /*  INITIALIZE COMMANDS  */
            FLAG=1;  /*  INITIALIZE BUFFER-FULL FLAG  */
        /*  FOLLOWING IS SEQUENCE OF INSTRUCTIONS WHICH ARE CYCLED  */
        /*  THROUGH CONTINUOUSLY  */
        BEGIN% DISABLE;  /*  DISABLE INTERRUPTS  */
        /*  CHECK FLAG TO SEE IF COMMAND BUFFER IS FULL  */
            IF FLAG=0
        /*  IF NOT, SKIP FOLLOWING PROCEDURE  */
              THEN DO;
                    FLAG=1;  /*  RESET FLAG  */
                    TIME= 0FFH;  /*  RESET TIMER  */
                    DS= BUF(0);  /*  GET DS# FROM BUFFER  */
        /*  CHECK FOR VALID DS#  */
                    IF((DS > 0) AND (DS < 4))
                      THEN DO;
        /*  ACCUMULATE SPEED AND RATE COMMANDS  */
```

```
*D MODCOMP SOURCE EDITOR     DATE  01/03/80  14:44:20        PAGE   4

                         SPD(DS) = SHL(DOUBLE(SHL(BUF(1),4)+BUF(2)),4)
                               +BUF(3);
                         PT(DS) = SHL(DOUBLE(SHL(BUF(4),4)+BUF(5)),4)
                               +BUF(6);
                        END;
                 CALL ECHO(DS);  /*  ECHO TO MODCOMP  */
     END;
     ENABLE;  /*  RE-ENABLE INTERRUPTS  */
     GO TO BEGIN;  /*  LOOP BACK  */
     /*                                                     */
     /*                                                     */
     /*  NOTE:  IN IMPLEMENTING THIS DESIGN, IT WAS FOUND THAT A  */
     /*     DELAY WAS NEEDED BEFORE ECHOING TO THE MODCOMP.  THE  */
     /*     INSTRUCTIONS TO ACHIEVE THIS DELAY WERE ADDED DIRECTLY  */
     /*     TO THE EPROM.  THEREFORE, THE ACTUAL PROGRAM STORED IN  */
     /*     THE EPROM IS SLIGHTLY DIFFERENT FROM THE ONE PRESENTED */
     /*     HERE  */
     EOF
```

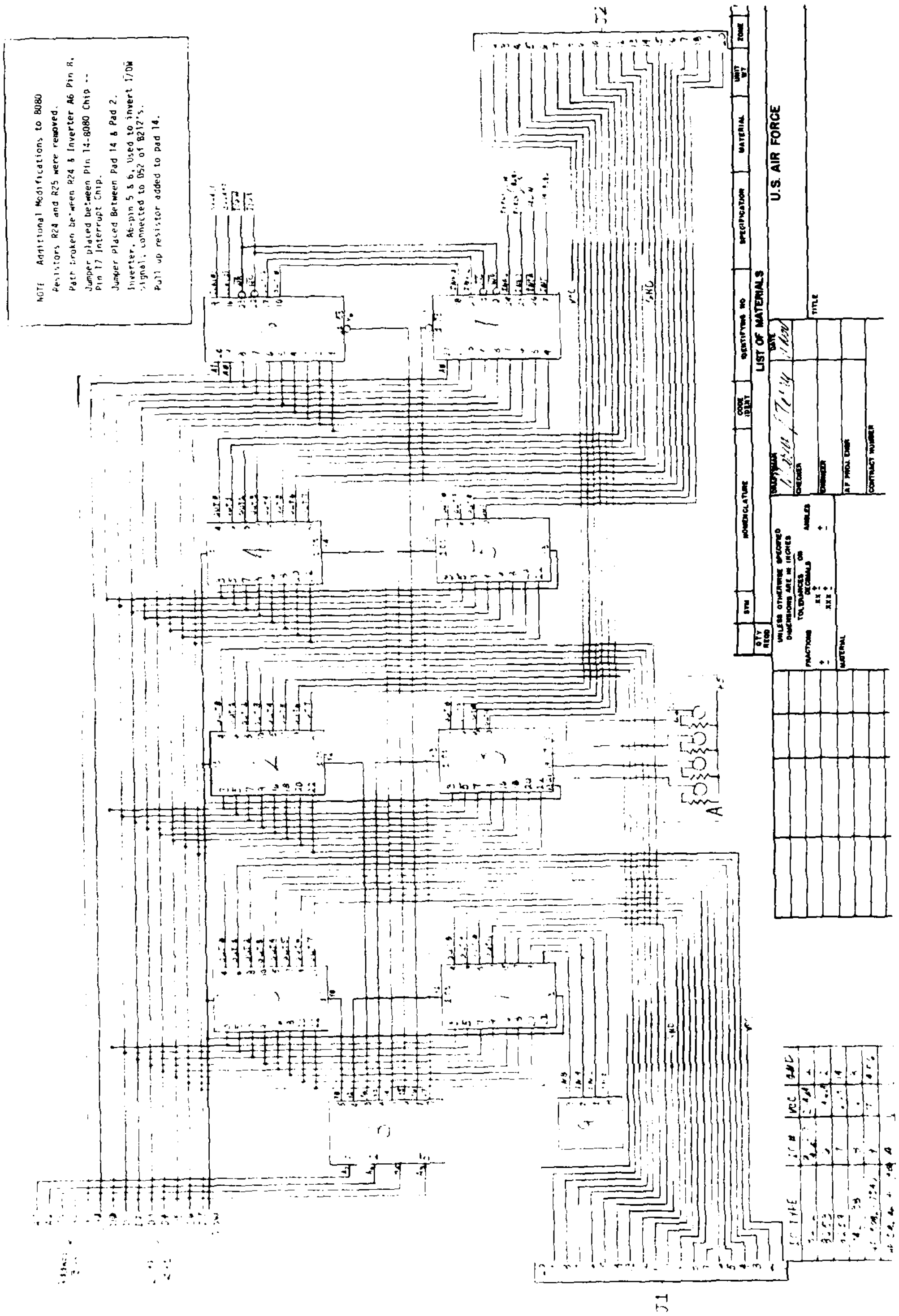

Figure E-1. Drive Stand Controller

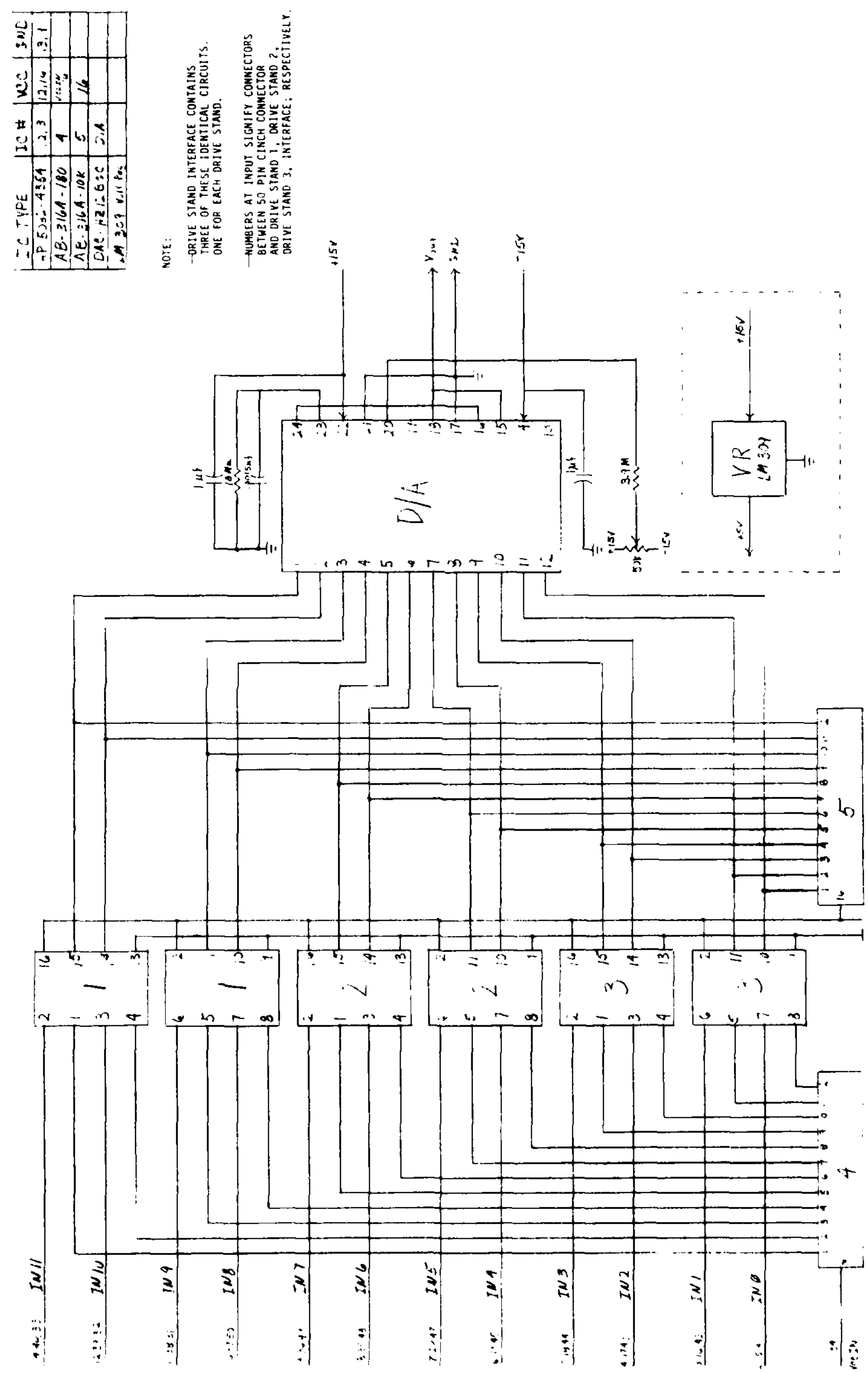

Figure E-2. D/A Converter

APPENDIX F

SAFETY CONDITION MONITORING SYSTEM

One feature of the Generator Test Facility is the automatic monitoring of certain safety conditions during computer-controlled testing. This safety monitoring is implemented using: (1) signals input to the minicomputer via the wide range analog input system, (2) a direct memory processor (DMP) routine, INITSAF, which stores these inputs in a dedicated buffer, and, (3) an overlimit checking routine, SAFCHECK, which executes during test sequence operation. The checking routine has output routines which, in the event of an overlimit safety condition, take actions to shut down the generator test. Following is a detailed description of these features.

In order to monitor a condition for safety purposes, a DC voltage signal proportional to that condition must be developed. This voltage signal should be relatively smooth and can be in the range from a few millivolts to ten volts. However, for better noise rejection and in the interest of standardization, it is recommended that a signal varying from 0-5 VDC be supplied. In addition, filtering and/or amplification of the raw transducer output is sometimes required.

After conditioning, these voltage signals are presented to an input channel of the analog input system of the Modcomp minicomputer. The analog input system accepts voltage inputs of 12 different ranges varying from 5 millivolts to 10.24 volts. The software routine used to input from the device selects the range for reading and optionally a zero suppression value. The zero suppression feature measures the variance of a reading about a preselected value. By specifying the safety limit for each signal as its zero suppression value, a positive input is present only if the signal is over its limit.

The analog input system can address up to 128 different channels. The channel addresses which exist in the system depend upon which card slots in the card file are occupied. Refer to Modcomp technical manual for a full description of the wide-range solid state analog input system.

Following is a description of the software routines used to input, store, and check the safety condition signals. These specific routines implement a safety monitoring scheme used for a 30/40 KVA VSCF generator system. To provide a safety monitoring scheme for other systems, these routines must be modified to reflect the new system's limits and particular safety signals. The signals monitored and limits used for the example safety scheme are given in Figure F-1.

The subroutine INITSAF initiates input from the analog input system. INITSAF is called by the routine RUNSYS immediately before execution of any test sequence is begun. Input is performed by a DMP which leaves the main processor of the minicomputer free to control generator test sequence execution. The DMP input causes a dedicated memory buffer to be filled with indications of any overlimit safety condition. All that is required to complete the safety monitoring scheme is a method to recognize an over-limit signal and the means to shut the test down. These features are included in the drive stand control task, DSC.

The drive stand control task is the software routine which controls the drive stand speed and acceleration rate. This control is provided through an Intel 8080 microprocessor-based conditioning circuit. Details of the control software of the DSC task and the drive stand controller are given in other sections of this manual. What is important to the safety monitoring scheme is that the DSC task is executing almost continuously during the execution of a generator test sequence. Therefore, by using the DSC task to invoke the overlimit checking routine, a constant monitoring of safety conditions is provided.

Upon activation of the DSC task, a key in the task communication area is set to zero. As long as this key (identified as SBF) is zero, no overlimit checking is performed. The RUNSYS routine, upon initiation of execution of a test sequence, sets the SBF key to a nonzero value. In fact, the SBF key is set to the address of the buffer of safety data located in central memory. Setting the SBF key causes the SAFCHECK subroutine of the DSC task to perform an overlimit check on each condition being monitored. Because of the zero suppression scheme employed by the INITSAF routine, any positive value in the safety buffer represents an overlimit signal.

AIS CHANNEL #	SIGNAL	LIMIT	TRANSDUCER OUTPUT
4	DRIVE STAND SPEED	8667 RPM	8.667 V
0	OIL TEMPERATURE IN	75°C	75 mV
1	OIL PRESSURE IN	120 psi*	-1200 mV
5	VIBRATION	8G	400 mV
68	DRIVE STAND BEARING #1 TEMPERATURE	100°F	2.56 V
69	DRIVE STAND BEARING #2 TEMPERATURE	100°F	2.56 V
70	DRIVE STAND BEARING #3 TEMPERATURE	100°F	2.56 V
71	DRIVE STAND BEARING #4 TEMPERATURE	100°F	2.56 V
64	GEARBOX BEARING #1 TEMPERATURE	200°F	4.0 V
65	GEARBOX BEARING #2 TEMPERATURE	200°F	4.0 V
66	GEARBOX BEARING #3 TEMPERATURE	200°F	4.0 V
67	GEARBOX BEARING #4 TEMPERATURE	200°F	4.0 V
7	DRIVE STAND SPEED	3000 RPM*	3 V

* OIL PRESSURE IN must be above 120 psi when the drive stand speed monitored at channel #7 is above 3000 RPM

Figure F-1 - Example of Safety Scheme

In the event a safety condition is found to be overlimit, the SAF-CHECK routine takes the particular shutdown action for that condition. In this way, the most critical actions for that condition can be performed. The mechanism for performing shutdown involves basically closing a relay contact in the minicomputer input/output interface subsystem which controls power to the circuit which performs the action. The SAF-CHECK routine also issues a message to the test operator on the ISC color display terminal. This message states what condition caused the shutdown and what actions were taken during shutdown. Also during shutdown, the DSC task puts itself in a hold state. As explained in Appendix E, this forces the drive stands to decelerate to zero speed.

To continue testing after an automatic shutdown, the test operator must take several steps. First, he must discover and correct the conditions which caused the shutdown. After correcting the overlimit condition, the operator may have to perform other steps. If the message, "DRIVE POWER REMOVED" is issued on the ISC display, the operator must disable the "lock-out" mechanism of the drive stand controller. This lockout interrupts the DC voltage supplied to the drive stand power supply. To reset the lockout, momentarily depress the red push button on the top of the lockout box. Specific shutdown procedures may require additional restart actions. These will be included in the ISC message. Finally, the DSC task must be resumed. This is done by depressing the CONSOLE INTERRUPT switch on the Modcomp minicomputer control panel and typing/DSC/A on the teletype console. This aborts the DSC task which not only resumes its operation but also causes the safety checking to be inactive until the RUNSYS routine activates it again.

It should be noted that the RUNSYS routine continues executing during and after an automatic shutdown. This execution must be allowed to complete before any restart actions are performed, so that all actions initiated during the test sequence are completed. When RUNSYS has completed a test sequence and is ready for a new test sequence maintenance command, it issues the message "ENTER COMMAND" on the Tektronix CRT screen. The test restart actions can then be performed.

Copies of the code for the INITSAF and SAFCHECK routines are included at the end of this appendix. These routines are written in Modcomp Fortran and assembly language.

Also included is a copy of the routine SAFETY. This routine can be run when operating the generator test facility in the manual mode. SAFETY inputs the safety data which is processed by the INITSAF and SAFCHECK routines during computer-controlled testing. However, instead of checking for overlimit conditions, SAFETY presents a tabular display of the signals on the ISC display screen. Thus during manual operation, the test operator can visually monitor these safety conditions. In addition, the SAFETY routine should be run prior to computer-controlled testing to insure that the safety signal transducers, and the minicomputer analog input system are operating correctly.

To execute the SAFETY routine, the test operator should type the following commands on the Tektronix terminal:

```
[JOB]
[EXE SAF LM]
```

The routine will respond with the message,

```
ENTER MAX SPEED
```

The user then enters the maximum drive stand speed to be allowed during manual operation. The drive stand speed, in addition to being displayed on the ISC terminal, is also checked against this maximum speed. If the drive stand speed exceeds this maximum, the routine SAFETY activates an emergency trip of the drive stand which then coasts to zero speed. This protection is provided in the manual mode since generator overspeed is probably the most critical unsafe condition.

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  22:24:44        PAGE   1

      SUBROUTINE INITSAF(ITEMP)
C
C  ROUTINE INITIALIZES A DMP TRANSFER OF SAFETY MONITORING
C  DATA INPUT FROM ANALOG INPUT SYSTEM (DEVICE NAME=AII) TO
C  BUFFER (IBUF).
C  EACH ANALOG INPUT HAS A ZERO SUPPRESSION VALUE EQUAL TO THE
C  UPPER LIMIT OF THE SIGNAL BEING MONITORED, SO THAT AN
C  ENTRY IS MADE INTO THE BUFFER ONLY IF THE SIGNAL EXCEEDS
C  THE UPPER LIMIT.
C  THE DSC TASK SCANS THE BUFFER AND TAKES 'SHUTDOWN' ACTION
C  IF IT FINDS ANY NON-ZERO ENTRY.
C
C  NOTE:  ITEMP IS A LOCATION IN ROUTINE RUNSYS IN WHICH THIS
C  SUBROUTINE STORES ADDRESS OF ORIGINAL SYSTEM SI INTERRUPT
C  HANDLER FOR DEVICE AII.
C
C
C  ROUTINE CURRENTLY HANDLES 13 CHANNELS OF ANALOG INPUT;
C  MAXIMUM IS 16.
C
C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C  *  *  *  CATALOGUED ON UL 10,31,79   *  *  *  *  *  *  *  *
C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C
C
      DIMENSION ITAB(34)
      DATA ITAB/Z5404,Z0D8A,Z5400,Z001E,Z5401,ZFE20
     1,Z5405,Z0140,Z5444,Z0400,Z5445,Z0400,Z5446,Z0400
     1,Z5447,Z0400,Z5440,Z0640,Z5441,Z0640,Z5442,Z0640
     1,Z5443,Z0640,Z5407,Z04B0,8*0/
C
C  ITAB IS TABLE WHICH CONTROLS INPUT OF ANALOG DATA.
C  EACH CHANNEL IS CONTROLLED BY A TWO-WORD ENTRY.  FIRST
C  WORD SPECIFIES RANGE AND CHANNEL, SECOND WORD SPECIFIES
C  ZERO SUPPERSSION VALUE.
C  FOR COMPLETE DESCRIPTION OF ANALOG INPUT SYSTEM, REFER TO
C  MODCOMP INPUT/OUTPUT MANUAL.
C
C  SIGNALS PRESENTLY BEING MONITORED:
C
C  CHANNEL#      SIGNAL          RANGE          LIMIT
C      4       DS SPEED            A            8.667V=8667RPM
C      3       OIL TEMP IN         A            75MV=75C
C      1      OIL PRESS IN         A
C      7      D.S. SPEED           A
C      2      VIBRATION            A      400MV=8G
C     68      D.S. BEAR. TEMP      A
C     69      D.S  BEAR. TEMP      A
C     70      D.S  BEAR. TEMP.     A
C     71      D.S. BEAR. TEMP.     A
C     64      G.B. BEAR. TEMP.     A
C     65      G.B. BEAR. TEMP.     A
C     66      G.B. BEAR. TEMP.     A
C     67      G.B. BEAR. TEMP.     A
C
C
```

```
C
C  NCH=NUMBER OF CHANNELS BEING MONITORED
      NCH=13
      LNCH=NCH*2
C  FOLLOWING IS ASSEMBLY LANGUAGE CODE WHICH INITIATES DMP
C  TRANSFER
      INLINE
         LDM,10  #00D3      STORE SI TRAP ADDRESS IN TEMP. LOCATION
         STM*,10  ITEMP
         LDI,2   SI         SI INTERRUPT HANDLER
         STM,2   #00D3      PUT IN SI TRAP
         LDI,2   NCH        GET ADDR OF NCH
         LDX,5,2            R5=NCH
         TTR,2,5            R2=-NCH,INPUT WORD COUNT
         ZBR,2,0            SET DMP LINK BIT
         STM,2   #0066      INPUT TC
         STM,2,5 IBUF       1ST WORD AFTER DATA BUFFER
         LDI,2   LNCH       GET ADDR OF LNCH
         LDX,6,2            R6=LNCH
         TTR,2,6            R2=-LNCH,OUTPUT WORD COUNT
         ZBR,2,0            SET DMP LINK BIT
         STM,2   #0065      OUTPUT TC
         STM,2,6 ITAB       1ST WORD AFTER SCAN TABLE
         LDI,2   IBUF       ADDR. OF DATA BUFFER
         STM,2   #0076      INPUT TA
         ABR,5,15           R5=NCH+1
         STM,2,5 IBUF       2ND WORD AFTER DATA BUFFER
         LDI,2   ITAB       ADDR. OF SCAN TABLE
         STM,2   #0075      OUTPUT TA
         ABR,6,15           R6=LNCH+1
         STM,2,6 ITAB       2ND WORD AFTER SCAN TABLE
*  USE CUSTOM REX CALL TO PASS ADDRESS OF DATA BUFFER, IBUF,
*  TO DSC TASK
         LDI,2   @SBF       SBF=SAFETY BUFFER
         LDI,3   IBUF
         REX,#4E
*  ALSO PASS NUMBER OF CHANNELS OF DATA, NCH, TO DSC TASK
         LDI,2  @NCH
         LDI,3  NCH
         LDX,3,3
         REX,#4E
*  INITIATE DMP
         LDI,3  #4400      SET UP TERMINATE COMMAND
         OCB,3,2           TERMINATE OUTPUT
         OCB,3,3           TERMINATE INPUT
BSY1     ISB,3,3     GET STATUS OF INPUT
         TBRB,3,7 BSY1     IF BUSY, LOOP
BSY2     ISB,3,2     GET STATUS OF OUTPUT
         TBRB,3,7 BSY2     IF BUSY, LOOP
         LDI,3  #C000      SET UP TRANSFER INITIATE COMMAND,IMP,DI=0
         OCB,3,3           INITIATE INPUT
         OCB,3,2           INITIATE OUTPUT
         BRU  OUT          JUMP AROUND DATA
SI       CIR             IGNORE SI INTERRUPTS
IBUF     RES  18,0
OUT      NOP
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  22:24:44          PAGE

          FINI
          RETURN
          END
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  22:25:03          PAGE   1

      SUBROUTINE SAFCHECK
C
C  ROUTINE WHICH CHECKS SAFETY MONITORING BUFFER FOR
C  ANY ENTRY LARGER THAN LIMIT
C  CURRENT ROUTINE USED FOR F-18 VSCF TEST
C
C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C  *  *  *  CATALOGUED ON UL  11,9,79  *  *  *  *  *  *  *  *  *  *
C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C
C  EXTERNAL SUBROUTINES REQUIRED:
C
C   NAME          LOCATION
C
C  1. AIFL           LB
C  2. INITISC        UL
C  3. ALNUM          UL
C  4. SETBF          UI
C  5. SETBT          UL
C  6. BLNKOFF      UL
C
C
C  INITIALIZE STORAGE LOCATIONS
      ITEMP1=0
      ITEMP2=0
C  GET ADDRESS OF SAFETY BUFFER (SBF) FROM TASK DATA AREA
      INLINE
        LDI,2  @SBF
        REX,#4D
        STM,3  ITEMP1
      FINI
C  IF SBF ADDR. = 0, SAFETY ACQUISITION HAS NOT BEEN INITIATED
      IF(ITEMP1.EQ.0)RETURN
C  GET NUMBER OF CHANNELS OF SAFETY DATA BEING USED (NCH)
      INLINE
        TRR,4,3     SAVE ADDRESS OF SBF IN REG 4
        LDI,2  @NCH
        REX,#4D
        STM,3  ITEMP1
      FINI
C  CHECK FOR ANY VALUE OVER LIMIT
C  NOTE:  LOOP EXTENDS ONLY TO NCH-1, SINCE LAST CHANNEL
C  IS A MINIMUM DS SPEED WHICH IS CHECKED ONLY IF OIL PRESS IS LOW
      JEND=ITEMP1-1
      DO 10 J=1,JEND
C  GET ONE WORD OF SAFETY DATA FROM SBF
      INLINE
        LDX,2,4     REG 2 = SAFETY WORD
        STM,2  ITEMP2     STORE IN FORTRAN VARIABLE
      FINI
C  CONVERT ANALOG INPUT VALUE TO DECIMAL
      CALL AIFL(1,ITEMP2,DATA)
C  CHECK FOR OVER LIMIT VALUE
      IF(DATA.GT.0.)GO TO 50
C  INCREMENT POINTER INTO SBF
      INLINE
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  22:24:44          PAGE   7

          FINI
          RETURN
          END
```

*D MODCOMP SOURCE EDITOR   DATE 12/27/79 22:25:03   PAGE   2

```
            ABR,4,15
          FINI
10        CONTINUE
C  IF NO "OVER LIMIT" VALUES, RETURN
          RETURN
C  ERASE ISC SCREEN WITH BLINK ON
C  SELECT PROPER SHUTDOWN ROUTINE
50        GO TO(100,300,400,500,700,700,700
         1,700,800,800,800,800),J
C  ROUTINE TO HANDLE DRIVE STAND OVER SPEED
100       CALL STOPDRIVE
          CALL EMERTRIP
          CALL INITISC(1)
C  FOLLOWING STATEMENTS PRINT MESSAGE ON ISC DISPLAY
          CALL ALNUM(2,10,'DS OVERSPEED',12)
          CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
          CALL ALNUM(2,20,'DS EMERGENCY TRIP',18)
          CALL BLNKOFF
C  HALT DSC TASK AND PRINT MESSAGE
C         INLINE
C           REX,#90
C           DFC MES1
C           BRU OUT
CMES1        DFC " DS OVERSPEED",0
COUT         NOP
C         FINI
          CALL BELL
          RETURN
C  ROUTINE TO HANDLE REVERSE DRIVE STAND ROTATION
200       CALL STOPDRIVE
          CALL EMERTRIP
          CALL INITISC(1)
C  ISC MESSAGE
          CALL ALNUM(2,10,'REVERSE DS ROTATION',20)
          CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
          CALL ALNUM(2,20,'DS EMERGENCY TRIP',18)
          CALL ALNUM(2,25,'DSC TASK HALTED',16)
          CALL BLNKOFF
          INLINE
            REX,#90
            DFC MES2
            BRU OUT2
MES2        DFC " REVERSE DS ROTATION",0
OUT2        NOP
          FINI
          RETURN
C  OIL TEMPERATURE ROUTINE
300       CALL STOPDRIVE
C  ISC MESSAGE
          CALL INITISC(1)
          CALL ALNUM(2,10,'OIL TEMP IN TOO HIGH',20)
          CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
          CALL ALNUM(2,20,'DSC TASK HALTED',16)
          CALL BLNKOFF
          INLINE
            REX,#90
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  22:25:03        PAGE   3

               DFC MES4
               BRU OUT4
     MES4      DFC " OIL TEMP HI",0
     OUT4      NOP
           FINI
           RETURN
     C  OIL PRESSURE ROUTINE
     400   CONTINUE
     C  CHECK DRIVE STAND SPEED
           INLINE
               ABR,4,14
               ABR,4,12
               LDX,2,4
               STM,2 ITEMP2      MIN. DS SPEED
           FINI
     C  CHECK IF DS UP TO SPEED
           CALL AIFL(1,ITEMP2,DATA)
           IF(DATA.LT.0.)RETURN
           CALL STOPDRIVE
           CALL INITISC(1)
     C  ISC MESSAGE
           CALL ALNUM(2,10,'LOW OIL PRESSURE',16)
           CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
           CALL ALNUM(2,20,'DSC TASK HALTED',24)
           CALL BLNKOFF
           INLINE
               REX,#90
               DFC MES5
               BRU OUT5
     MES5      DFC " LOW OIL PRESSURE",0
     OUT5      NOP
           FINI
           RETURN
     C  VIBRATION ROUTINE
     500   CALL STOPDRIVE
           CALL INITISC(1)
     C  ISC MESSAGE
           CALL ALNUM(2,10,'EXCESSIVE VIBRATION',20)
           CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
           CALL ALNUM(2,20,'DSC TASK HALTED',16)
           CALL BLNKOFF
           INLINE
               REX,#90
               DFC  MES6
               BRU  OUT6
     MES6      DFC   " EXCESSIVE VIBRATION",0
     OUT6      NOP
           FINI
           RETURN
     C  SCAVENGE PUMP ROUTINE
     600   CALL STOPDRIVE
     C  ISC MESSAGE
           CALL ALNUM(2,10,'SCAVENGE NOT OPERATING',22)
           CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
           CALL ALNUM(2,20,'DSC TASK HALTED',16)
           CALL BLNKOFF
```

```
        INLINE
          REX,#90
          DFC  MES7
          BRU OUT7
MES7      DFC  " SCAVENGE NOT OPERATING",0
OUT7      NOP
        FINI
        RETURN
C  DRIVE STAND BEARINGS ROUTINE
700     CALL STOPDRIVE
        CALL INITISC(1)
C  ISC MESSAGE
        CALL ALNUM(2,10,'DS BEARINGS OVERHEATED',22)
        CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
        CALL ALNUM(2,20,'DSC TASK HALTED',16)
        CALL BLNKOFF
        INLINE
          REX,#90
          DFC  MES8
          BRU  OUT8
MES8      DFC  " DRIVE STAND BEARINGS OVERHEATED",0
OUT8     NOP
        FINI
        RETURN
C  GEAR BOX BEARINGS ROUTINE
800     CALL STOPDRIVE
        CALL INITISC(1)
C  ISC MESSAGE
        CALL ALNUM(2,10,'GEAR BOX BEARINGS TOO HOT',26)
        CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
        CALL ALNUM(2,20,'DSC TASK HALTED',16)
        CALL BLNKOFF
        INLINE
          REX,#90
          DFC  MES9
          BRU OUT9
MES9      DFC  " GEAR BOX BEARINGS OVERHEATED",0
OUT9      NOP
        FINI
        RETURN
        END
C
C
        SUBROUTINE SETBIT(NBIT)
C
C  ROUTINE TO SET BIT AND OUTPUT IN EXTERNAL RELAY
C  OUTPUT SCHEME
C
        DATA IWORD/0/,NSCAN/Z4019/
C  SET BIT IN CONTROL WORD
        CALL SETBF(IWORD,NBIT)
C  OUTPUT CONTROL WORD
        INLINE
          LDI,2 UFT
          REX,1
          DFC IWORD,2
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  22:25:03        PAGE   5

               BRU OUT
               DFC NSCAN,1
     UFT       DFC 0,@D00,#8400,2,0,0
     OUT       NOP
            FINI
            RETURN
            END
     C
     C
            SUBROUTINE CLRBIT(NBIT)
     C
     C   ROUTINE TO CLEAR BIT AND OUTPUT IN EXTERNAL RELAY OUTPUT
     C   SCHEME
     C
            DATA IWORD/0/,NSCAN/Z4019/
     C   CLEAR BIT IN CONTROL WORD
            CALL SETBT(IWORD,NBIT)
     C   OUTPUT CONTROL WORD
            INLINE
               LDI,2 UFT
               REX,1
               DFC IWORD,2
               BRU OUT
               DFC NSCAN,1
     UFT       DFC 0,@D00,#8400,0,0,0
     OUT       NOP
            FINI
            RETURN
            END
     C
     C
            SUBROUTINE STOPDRIVE
     C
     C   FIRST USED IN 150 KVA IDG TEST
     C   ALSO USED IN F-18 VSCF TEST
     C   SHOULD BE APPLICABLE FOR ALL TESTS
     C   PUTS PULSE ON CONTACT #12, WHICH REMOVES DS CONTROL
     C   VOLTAGE AND ACTIVATES "LOCK-OUT"
     C   "LOCK-OUT" IS RESET WITH RED BUTTON ON 8080 DS CONTROL
     C INTERFACE BOX
     C
     C   SET BIT
            CALL SETBIT(4)
     C   WAIT .5 SEC
            INLINE
               REX,#14
               DFC #8000,100
            FINI
     C   CLEAR BIT
            CALL CLRBIT(4)
            RETURN
            END
     C
     C
            SUBROUTINE EMERTRIP
     C
```

```
C   USED FIRST IN 150 KVA IDG TEST
C   SHOULD BE APPLICABLE FOR ALL TESTS
C   PUTS A PULSE ON CONTACT #7, WHICH ACTIVATES DS EMERGENCY
C   TRIP
C
C   SET BIT
          CALL SETBIT(12)
C   WAIT .5 SEC
          INLINE
            REX,#14
            DFC #8000,100
          FINI
C   CLEAR BIT
          CALL CLRBIT(12)
          RETURN
          END
C
C
          SUBROUTINE BLNKOFF
          INLINE
            LDI,2  UFT
            REX,1
            DFC   BUF,2
            BRU   OUT
UFT         DFC   0,@ISC,#9080,0,0,0
BUF         DFC   #0F0E
OUT         NOP
          FINI
          RETURN
          END
C
C
          SUBROUTINE BELL
1         CONTINUE
          INLINE
            LDI,2  UFT
            REX,1
            DFC   BUF,2
            BRU OUT
UFT         DFC   0,@ISC,#9080,0,0,0
BUF         DFC   #0707
OUT         NOP
          FINI
          IF(1.EQ.2)RETURN
          GO TO 1
          END
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  22:25:27          PAGE   1

            PROGRAM SAFETY
      C
      C  THIS ROUTINE IS RUN WHEN OPERATING A GENERATOR IN THE MANUAL MODE.
      C  IT SERVES TWO FUNCTIONS.  FIRST, ALL SAFETY SIGNALS BEING MONITORED
      C  ARE DISPLAYED IN TABULAR FORM.  THIS DEMONSTRATES THAT THOSE AIS
      C  CHANNELS,TRANSDUCERS,ETC. ARE FUNCTIONING PROPERLY BEFORE COMPUTER
      C  CONTROLLED TESTING IS BEGUN.  NOTE THAT FOR EACH NEW GENERATOR
      C  SYSTEM, THIS ROUTINE MUST BE CHANGED TO TAKE INTO CONSIDERATION
      C  ALL SAFETY SIGNALS USED FOR THAT SYSTEM.  SECONDLY, THE USER ENTERS
      C  THE MAXIMUM DRIVE STAND SPEED ALLOWABLE FOR THE CURRENT GENERATOR.
      C  IF THIS SPEED IS EXCEEDED, THE EMERGENCY TRIP CIRCUIT IS ACTIVATED,
      C  WHICH STOPS THE DRIVE STAND.
      C
            DIMENSION INUFT(22),IBUF(12)
            DATA INUFT/0,12,0,0,Z8400,0,0,0,'AI','I',Z0204
           1,Z0200,Z0201,Z0205,Z0244,Z0245,Z0246,Z0247
           1,Z0240,Z0241,Z0242,Z0243/
      C  USER ENTERS MAX SPEED
            WRITE(8,100)
      100   FORMAT(' ENTER MAX SPEED')
            READ(7,200)SPEED
      200   FORMAT(F10.2)
      C  ERASE ISC SCREEN
            CALL GRAPHICS(3)
            CALL ERASCREEN
      10    CONTINUE
      C  LOOP TO INPUT AND DISPLAY SAFETY SIGNALS
            DO 70 KK=1,10000
      C  INPUT BUFFER OF SAFETY DATA
            CALL AIRDW(12,INUFT,IBUF,M)
      C  CONVERT DRIVE STAND SPEED TO FLOATING POINT
            CALL AIFL(1,IBUF(1),DATA)
      C  CHECK FOR OVERSPEED
            IF(DATA.LT.SPEED)GO TO 20
      C  ACTIVATE DRIVE STAND EMERGENCY TRIP
            CALL EMERTRIP
      C  OUTPUT MESSAGE
            CALL INITISC(1)
            CALL ALNUM(2,10,'DS OVERSPEED',12)
            CALL ALNUM(2,15,'DS EMERGENCY TRIP',18)
            CALL ALNUM(2,20,'SAFETY MONITORING HALTED',24)
      C  EXIT THIS ROUTINE
            STOP
      C  OUTPUT DRIVE STAND SPEED
      20    DATA=DATA*3.
            WRITE(6,300)DATA
      300   FORMAT(//' DRIVE STAND SPEED = ',F10.2,' RPM')
      C  CONVERT OIL TEMP TO FLT. PT.
            CALL AIFL(1,IBUF(2),DATA)
      C  OUTPUT OIL TEMP
            WRITE(6,400)DATA
      400   FORMAT(//' OIL TEMP IN = ',F10.2,' DEG C')
      C  CONVERT OIL PRESSURE TO FLT. PT.
            CALL AIFL(1,IBUF(3),DATA)
      C  CONVERT TO PSI
            DATA=DATA/10.
```

```
*D MODCOMP SOURCE EDITOR      DATE  12/27/79  22:25:27         PAGE   2

C   OUTPUT OIL PRESSURE
        WRITE(6,500)DATA
500     FORMAT(//' OIL PRESSURE IN = ',F10.2,' PSI')
C   CONVERT VIBRATION READING TO FLT. PT.
        CALL AIFL(1,IBUF(4),DATA)
C   CONVERT TO G'S
        DATA=DATA/100.
C   OUTPUT VIBRATION
        WRITE(6,600)DATA
600     FORMAT(//' VIBRATION = ',F10.2,' G')
C   CHECK BEARING TEMPERATURES
        WRITE(6,650)
650     FORMAT(80X)
C   INITIALIZE FLAG WHICH INDICATES HOW MANY OVER-TEMP BEARINGS
        NGOOD=0
C   CHECK DRIVE STAND BEARINGS
        DO 30 K=1,4
C   CONVERT TO FLOATING POINT
        CALL AIFL(1,IBUF(K+4),DATA)
C   CHECK FOR VALID READING
        IF(DATA.GT.500..AND.DATA.LT.2560.)GO TO 30
C   OUTPUT INVALID READING
        WRITE(6,700)K,DATA
700     FORMAT(' DRIVE STAND BEARING #',I1,' = ',F10.2,10X)
C   INCREMENT FLAG
        NGOOD=NGOOD+1
30      CONTINUE
C   CHECK GEAR BOX BEARINGS
        DO 40 K=1,4
C   CONVERT TO FLOATING POINT
        CALL AIFL(1,IBUF(K+8),DATA)
C   CHECK FOR VALID READING
        IF(DATA.GT.2500..AND.DATA.LT.3500.)GO TO 40
C   OUTPUT INVALID READING
        WRITE(6,800)K,DATA
800     FORMAT(' GEAR BOX BEARING #',I1,' = ',F10.2,10X)
C   INCREMENT FLAG
        NGOOD=NGOOD+1
40      CONTINUE
C   WERE ANY BEARING TEMPS BAD?
        IF(NGOOD.EQ.0)GO TO 60
C   IF THERE WERE ANY BAD READINGS, WIPE OUT "OLD" MESSAGES
        LPEND=9-NGOOD
        DO 50 LP=1,LPEND
50      WRITE(6,900)
900     FORMAT(80X)
C   GO TO END OF LOOP
        GO TO 70
C   OUTPUT MESSAGE INDICATING ALL BEARING TEMPS ARE O.K.
60      WRITE(6,1000)
1000    FORMAT(80X,/,80X,/
       1,' ALL DRIVE STAND AND GEAR BOX BEARING'
       1,/,' TEMPERATURES ARE WITHIN LIMITS',10X,4(/,80X))
C   RESET CURSOR TO REPRINT TABLE
70      CALL HOME
C   ALLOW USER TO RESTART LOOP OR QUIT
```

*D MODCOMP SOURCE EDITOR   DATE 12/27/79 22:25:27   PAGE   3

```
           WRITE(8,1100)
 1100      FORMAT(' IS TEST COMPLETE?(Y OR N)')
           READ(7,1200)IANS
 1200      FORMAT(A1)
           IF(IANS.EQ.'N')GO TO 10
           STOP
           END
C
C
           SUBROUTINE HOME
C
C   ROUTINE WHICH MOVES CURSOR OF ISC TERMINAL TO HOME POSITION.
C
           INLINE
             LDI,2  UFT
             REX,1
             DFC   BUF,2
             BRU   OUT
UFT           DFC   0,@ISC,#9030,0,0,0
BUF           DFC   #0800
OUT           NOP
           FINI
           RETURN
           END
```


☆U.S.Government Printing Office: 1980 — 657-084/697